

RAPPORT DE TRAVAIL DE BACHELOR

Commande d'un robot d'arrosage pour agriculture durable

Analyse d'image, robotique autonome et agriculture durable

Diplomant :
Ludovic RICHARD

Professeur :
Etienne MESSERLI

Mandant :
Christien BOVINI

Expert :
Philippe AMEZ-DROZ

HEIG-VD
Département TIC - Informatique Embarquée
Institut REDS
(Reconfigurable & Embedded Digital Systems)

30 juillet 2018

TRAVAIL DE BACHELOR 2017 - 2018

Commande pour robot d'arrosage pour agriculture durable

Domaine de recherche : Robotique autonome, agriculture durable

Entreprise Ferme bio du Moulin (Bovigny Christian)

Instituts COMATEC
REDS

Énoncé

L'objectif du projet est de réaliser un robot d'arrosage pour des plantons dans le but de soutenir une agriculture écologique et durable. Durant les premiers jours, voir les premières semaines, les plantons sont très sensibles à un manque d'eau par le faible volume et profondeur de leurs racines. Ce stress peut engendrer des pertes de production assez conséquente. L'objectif est de développer un robot autonome permettant d'assurer l'arrosage plant par plant de façon régulière tout en maîtrisant le volume d'eau délivré, ce qui permettrait des économies d'eau.

Le projet de robot d'arrosage est décomposé en deux propositions de projets de TB. Un premier TB traitera de la réalisation mécanique du robot avec les actuateurs, tandis que le second s'occupera de la commande de celui-ci.

Le présent projet concerne la commande du robot d'arrosage. Celle-ci doit permettre le déplacement autonome du robot, la détection de la position des plantons et le pilotage du système d'arrosage. Elle doit être embarquée sur le robot et permettre le contrôle à distance. Il est imaginé que le système de commande est basé sur une plateforme embarquée, qui dispose de fonction de géo-localisation (GPS), de moyen de communication et de possibilité de stockage des séquences réalisées.

Le système d'arrosage doit pouvoir être utilisé dans la configuration actuelle des plantations réalisées par le maraîcher. Il cultive ces légumes en bande de 150cm de large (entre axe du tracteur) sur 120 à 150 m de long. Celles-ci comprennent 3 lignes de plantons espacées d'environ 40 cm. La distance entre les plantons est de 30cm et chaque ligne est décalée.

Robotique, contrôle, vision, système embarqué, prog C

Cahier des charges

Un robot pour l'agriculture durable peut disposer de nombreuses fonctionnalités. Pour ce premier projet réalisé sur cette commande de robot, nous avons planifié les principales

fonctionnalités. Il est à mentionner que le robot, partie mécanique, sera disponible uniquement en fin de projet car le projet correspondant se déroule en parallèle à celui-ci.

Voici les étapes prévues pour ce travail:

- Etablir les spécifications complètes du robot d'arrosage en collaboration avec l'étudiant réalisant la partie mécanique.
- Choix de la plateforme et des différents composants pour la commande embarquée du robot d'arrosage.
- Concevoir et développer la détection des plantons avec une ou plusieurs caméras
- Concevoir et développer la commande du ou des bras comportant une tête d'arrosage pour arroser les plantons détectés pendant l'avancement du robot.
- Concevoir et développer le programme de contrôle permettant de le déplacement autonome du robot dans un ligne de plantation des plantons. Le robot dispose de deux roues avec un moteur indépendant pour le diriger.
- Mettre en œuvre les infrastructures de test pour valider les différentes étapes.
- Organiser et réaliser un test dans le champ de l'agriculteur pour valider le fonctionnement du robot d'arrosage dans une ligne un ligne de plantation des plantons.

Bibliographie

<https://bovigny.ch/ferme-bio-du-moulin/>

Candidat

Richard Ludovic Date: _____ Signature: _____

Responsable

Messerli Etienne Date: _____ Signature: _____

Chef du département TIC

Sanchez Eduardo Date: Yverdon-les-Bains, le 23.07.2018 Signature: 

Ferme bio du Moulin

Bovigny Christian Date: _____ Signature: _____

Table des matières

Cahier des charges	iii
Résumé	xi
1 Introduction	1
1.1 Contexte du projet	1
1.2 État initial du projet	1
1.3 Structure et contenu de ce rapport	1
2 Démarrage du projet	3
2.1 De nombreuses questions...	3
2.2 ... et de maigres informations	3
2.3 Choix de la mécanique	4
2.4 Choix de l'informatique	5
2.4.1 Acquisition d'image	5
2.4.2 Spécifications	5
Fonctionnalités minimales	6
Fonctionnalités souhaitables	6
Fonctionnalités futures	7
2.5 Et pour le reste....	7
2.5.1 Partie électronique	7
2.5.2 Prototype	7
2.6 Étapes du projet	7
2.6.1 Détecter et arroser les plantons	8
2.6.2 Suivre la bande de terre	8
2.6.3 Se repérer et se déplacer dans le champ	9
2.6.4 Planning	9
3 Matériel	11
3.1 Besoins du projet	11
3.2 Carte de système embarqué	11
3.2.1 Besoins	11
3.2.2 Existant	12
3.2.3 Choix	14
3.2.4 Démarrage de la carte et installation d'openCV	15
3.3 Caméra	15
3.3.1 Besoins	15
3.3.2 Existant	15
3.3.3 Choix	16
3.3.4 Tests & Résultats	17
Test 1 - Lumières allumées	17

	Test 2 - Lumière très faible	18
	Analyse des résultats et conclusion	19
3.4	Écran	19
3.4.1	Besoins	19
3.4.2	Existant	19
3.4.3	Choix	20
3.4.4	Test du Waveshare 10,1'	21
3.5	Carte GPS	21
3.5.1	Besoins	21
3.5.2	Existant	21
3.5.3	Choix	22
3.6	Carte GSM	22
3.6.1	Besoins	22
3.6.2	Existant	22
3.6.3	Choix	22
3.7	Carte d'extension PWM	23
3.7.1	Besoins	23
3.7.2	Choix	23
3.8	Synthèse : Ce qu'il faut encore faire au niveau hardware	23
4	Reconnaissance des plantons	25
4.1	Les différentes possibilités	25
4.2	Plantons ou mauvaises herbes?	26
4.2.1	Position GPS	26
4.2.2	Affiner les paramètres de détection	26
4.3	Bibliothèque choisie	26
4.4	Algorithme	26
4.4.1	Binarisation	27
	Format de la couleur	27
	Couleur verte et tolérances	28
4.4.2	Opérations morphologiques	29
4.4.3	Séparation des différents plantons	29
	Gestion de la perspective	30
4.4.4	Calcul des positions et élimination des objets trop rapprochés	30
4.5	Paramètres, options et optimisations	31
4.5.1	Source, fenêtres de sortie, résolution et taille d'affichage	31
4.5.2	Fourchette de couleurs détectées	32
4.5.3	Taille du noyau des opérations morphologique	32
4.5.4	Distances et tailles des objets	32
4.5.5	Acquisition de données	32
4.5.6	Logs et debug	33
4.6	Tests & Résultats	33
4.6.1	Test 1 - Vidéo du champ	33
4.6.2	Test 2 - Vidéo filmée en laboratoire	38
4.6.3	Conclusion des tests	42
4.6.4	Améliorations possibles du logiciel	42
4.7	Futurs tests et optimisations	42

5 Suivis d'une ligne dans un champ	45
5.1 Gestion des roues	45
5.1.1 Moteurs	45
5.1.2 Contrôle de la carte PWM	45
5.1.3 Déplacements manuels	46
Mode0 (mode "voiture")	46
Mode1 (mode "tourner sur place")	47
5.2 Pilote automatique	47
5.2.1 Reconnaissance de la direction à prendre	47
5.2.2 Autre moyens	48
5.2.3 Fin de la ligne	49
5.3 Tests, état actuel et travail restant	49
6 Généralités logiciels	51
6.1 Fonctionnement général	51
6.1.1 Structure logiciel	51
6.1.2 Rôle des threads et concurrence	52
6.2 Pré-requis	52
6.3 Compilations	52
6.4 Options et paramètres	53
6.5 Démarrer un logiciel automatique après le boot	53
6.5.1 Démarrage avec le système	53
6.5.2 Démarrage en mode utilisateur	54
6.6 Améliorations possibles	54
7 Prototype et banc de test	55
7.1 Une mauvaise nouvelle	55
7.2 Le prototype	55
7.2.1 le reds bot	55
7.2.2 Prototype du bras	57
7.2.3 Alimentation	57
7.3 Le banc de test	57
8 Conclusion	59
8.1 Synthèse	59
8.2 Futur du projet	60
8.3 Commentaires personnels	60
Bibliographie	63
9 Authentication	65
Table des figures	68

Résumé

Commande d'un robot d'arrosage pour agriculture durable

Analyse d'image, robotique autonome et agriculture durable

Ludovic RICHARD

Ce rapport présente le travail réalisé pour la conception de la commande d'un robot d'arrosage, permettant de grande économie d'eau ainsi qu'un gain de production agricole. Le mandant est un maraîcher suisse soucieux de l'impact écologique de son exploitation. Le robot aura la forme d'un petit tracteur, fonctionnant uniquement à l'électricité et devra suivre une ligne de plantons, les détecter, puis les arroser avec précision. A noter que la conception a été séparée en deux parties distinctes. D'une part, la conception mécanique et d'autre part, la conception informatique de la commande du robot. Ce travail traite donc cette 2ème partie. Les principales lignes d'études sont donc le choix du matériel, la programmation d'un système embarqué sur Raspberry PI, l'analyse d'image à l'aide de la très célèbre librairie *openCV*, la commande de différents types de moteurs et la réalisation d'un prototype permettant de montrer le bon fonctionnement des algorithmes conçus. Il présente aussi les défis rencontrés et fait le point sur les connaissances et les compétences acquises au travers de ce travail. Il montre ainsi certains points auxquelles il faut être attentifs lors de la réalisation de projet conséquents tel que celui-ci.

Chapitre 1

Introduction

1.1 Contexte du projet

A l'heure où l'écologie est au cœur des débats, une grande quantité d'idées, de projets et de solutions émergent du monde de l'ingénierie. L'objet de ce travail ne fait pas exception. La proposition de projet provient d'un paysan propriétaire d'une ferme bio et soucieux de limiter l'impact écologique de son exploitation.

L'objectif est de réaliser un robot qui assure l'arrosage de plantons dans leurs premières semaines de pousse. En effet, durant cette période, les plantons sont très sensibles à un manque d'eau par le faible volume et profondeur de leurs racines. Ce stress peut engendrer des pertes de production assez conséquentes. Ce robot devra donc arroser les plantons de manière régulière et avec la quantité d'eau adéquate. De plus, une grande partie de la terre du champ n'est pas atteinte par les racines des plantons. Avec un arroseur classique qui arrose l'entier du camp, une grande quantité d'eau est perdue. L'intérêt de ce robot est donc double ; Limiter drastiquement la consommation d'eau lors de l'arrosage, tout en donnant aux plantons de meilleures chances de survie dans leurs premières semaines en terre.

Ce travail de bachelor traite uniquement de la partie commande du système. La conception et la réalisation de la partie mécanique du robot fait l'objet d'un autre travail de bachelor qui a été réalisé en parallèle de celui-ci par Quentin Baball.

1.2 État initial du projet

Aucun travaux n'avaient été réalisés avant le début de ce travail et aucune contrainte particulière concernant la réalisation du robot n'a été imposées. Le projet jouit donc d'une grande liberté. Cependant, la partie mécanique étant conçue, par un autre institut, en parallèle de ce travail et non en préambule, une communication étroite sera nécessaire pour assurer un suivi et une cohérence entre les deux partis.

1.3 Structure et contenu de ce rapport

Les chapitres de ce rapport décrivent les différents travaux effectués et sont agencés de manière chronologique. Les tests effectués et leurs résultats sont directement inclus dans le chapitre correspondant. Les travaux restants/suivants ou les améliorations possibles sont aussi inclus à la fin des chapitre. La création du prototype et du banc de test fait par contre l'objet d'un chapitre distinct.

Voici un résumé des grosses étapes du projet décrites dans les différents chapitres de ce document :

Chapitre 2 - Démarrage du projet : Il s'agit ici de définir les spécifications exactes du robot, en collaboration avec le mandant et avec les concepteurs de la partie mécanique. Il faut ensuite établir, toujours avec la partie mécanique, une structure générale pour le robot qui servira de base pour la suite de sa conception. C'est aussi dans ce chapitre que le projet est découpé en plusieurs étapes concrètes et qu'un planning général est définit.

Chapitre 3 - Matériel : Ce chapitre décrit les choix matériels qui ont été faits et les premiers travaux de mise en œuvre de ce matériel.

Chapitre 4 - Reconnaissance des plantons : Ce chapitre décrit l'algorithme utilisé pour déterminer la position des plantons à partir de d'un flux vidéo. Il montre aussi les résultats de 2 tests de détection effectués

Chapitre 5 - Suivi d'une ligne dans un champ : Ce chapitre décrit les différentes possibilités pour suivre automatiquement la ligne de terre d'un champ et les choix qui ont été faits pour ce travail

Chapitre 6 - Généralités logiciels : Ce chapitre présente l'ensemble du projet software réalisé ainsi que son architecture. Il décrit la manière dont le projet est organisé et les différentes options logicielles disponibles ainsi que les pré-requis pour pouvoir compiler le projet. Il indique aussi les améliorations possibles au niveau du soft pour la suite du projet

Chapitre 7 - Prototype, banc de tests et résultats : Ce chapitre décrit le prototype réalisé, les différents tests effectués et leur conditions et montre les résultats obtenus. Il indique aussi certaines corrections qui ont été nécessaires.

Chapitre 8 - Conclusion : En conclusion, ce chapitre fait la synthèse de l'état actuel du projet et résume les principales étapes à venir en proposant des solutions pour la suite de son développement. Il fait aussi le point sur les réussites et les échecs du développement et contient aussi quelques commentaires personnels.

Chapitre 2

Démarrage du projet

2.1 De nombreuses questions...

Pour pouvoir définir le cahier des charge de la partie commande du robot, il a d'abord fallu définir clairement les opérations qu'il devrait effectuer ainsi que sa structure générale. Les 1ères semaines ont donc été riches en questionnement, en discussions et en propositions. Voici une liste non-exhaustives des questions qui ont été posées et étudiées entre les deux étudiants, leur professeur et le client :

- Quelle taille aura le robot ?
- A quelle vitesse doit-il pouvoir avancer ?
- Quel sera sa source d'énergie ?
- De combien de tête d'arrosage doit-il disposer ?
- Quelle quantité d'eau doit-il pouvoir embarquer ?
- Comment détecter les plantons pour les arroser ?
- Comment être sûr de n'arroser que les plantons ?
- Sur quel(s) type(s) de bras sera(ont) fixé(s) l'(es) arroseur(s) ?
- Quels type de moteur seront utilisé pour faire avancer le robot / diriger les bras ?
- Quel interface utiliser pour programmer/diriger le robot ou recevoir des informations de sa part ?
- Doit-il être 100% autonome en eau et en énergie ?
- Quel menace/problème pourrait-il survenir et comment les gérer ?
- Le robot a-t-il besoin d'une connexion à distance ?
- ...

2.2 ... et de maigres informations

Pour répondre à ces nombreuses questions, le client n'ayant pas vraiment d'idée définie sur le fonctionnement du robot. Nous avions donc une vraie liberté quand à la réalisation du projet. A vrai dire, les seules vraies contraintes définies concernaient les dimensions du champs. Le système d'arrosage devait pouvoir être utilisé dans la configuration des plantations réalisées par le maraîcher. Il cultive ces légumes en bande de 150cm de large (entre axe du tracteur) sur 120 à 150 m de long. Celles-ci comprennent 3 lignes de plantons espacées d'environ 40 cm. La distance entre les plantons est de 30cm et chaque ligne est décalée. Le nombre total de bande de terre n'était pas clairement défini. La figure 2.1 illustre les dimensions citées ci-dessus.

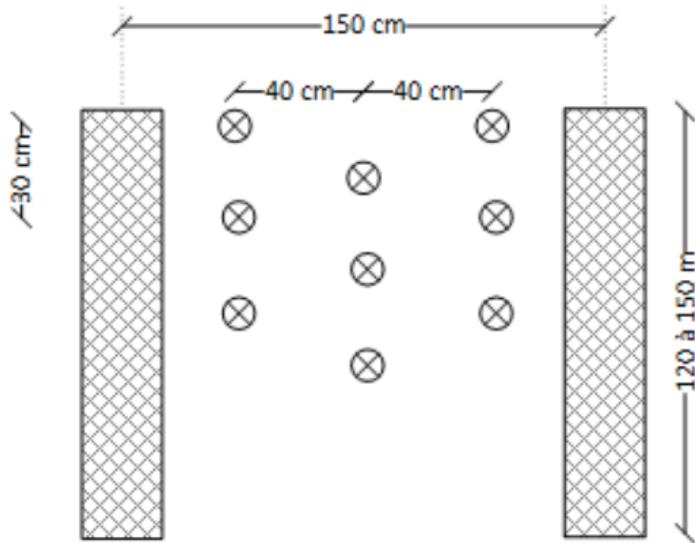


FIGURE 2.1 – Schémas du champs[1]

2.3 Choix de la mécanique

De nombreux choix ont dû être effectués et c'est tout d'abord l'aspect mécanique du robot qui a été défini. Lors de nos premières recherches et réflexions, nous avons découvert FarmBot[2]. Il s'agit d'un projet de robotique agricole déjà bien avancé et relativement semblable au nôtre. De plus, il est totalement open-source. Même si, au final, nous n'en avons rien tiré de vraiment profitable pour notre robot, il reste un projet très intéressant qui vaut le détour. Il me semblait donc important d'encourager ce genre de projet en en faisant mention dans ce rapport.

Voici les premières décisions qui ont été prises en collaboration avec l'étudiant en mécanique :

- Robot dans le style d'un mini-tracteur, de la largeur d'une bande (environ 1m50)
- Peut porter jusqu'à 300L d'eau dans des bidons fixés sur sa structure
- 2 roues motrices (2 moteurs) et 4 roues folles, avec système de stabilisation
- 3 bras (au lieu d'un seul) pour des questions de rapidité d'arrosage
- Si possible, le robot ne doit pas s'arrêter pour arroser
- Bras d'arrosage de type "coordonnées cylindriques", ce qui implique d'avoir 2 moteurs par bras et de gérer des angles
- Tous les moteurs se gèrent avec des PWMs

A ce stade du projet, la conception mécanique n'était pas plus clairement définie. Il a donc fallut spéculer sur les emplacements des bras et les possibilités de fixer des cartes et périphériques électroniques. Les modèles des moteurs n'étaient pas non plus encore clairement défini mais il était prévu déjà prévus qu'ils soient contrôlés par PWM (Pulse Width Modulation). La figure 2.2 montre le concept mécanique du robot final.

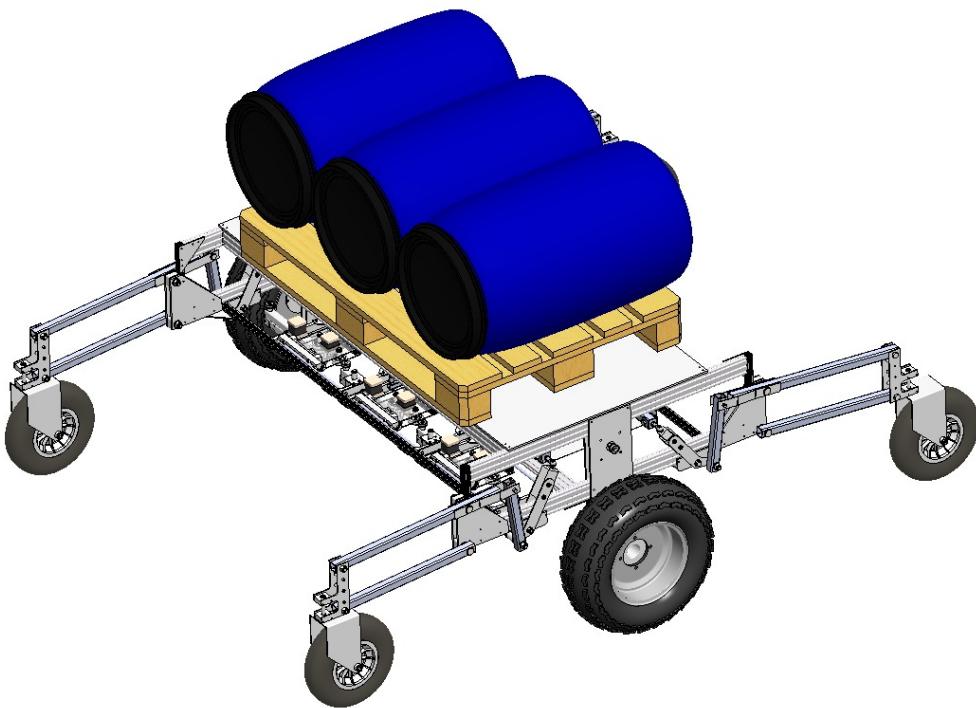


FIGURE 2.2 – Concept mécanique du robot[1]

2.4 Choix de l'informatique

Une fois ces informations mécaniques reçues, il a fallut définir les fonctionnalités électroniques et informatiques à implémenter.

2.4.1 Acquisition d'image

Le fait que le robot avaient besoin de camera pour se diriger et détecter les plantons semblait évident car tout autre type de capteurs auraient rendu la tâche bien plus ardue. Par contre, le nombre de caméra et leur emplacement sur le robot étaient plus difficile à déterminer. Deux possibilités ont alors été évoquées :

- **Une caméra sur le sommet du robot** prenant une vue d'ensemble permettant à la fois de suivre la ligne de planton et de surveiller la position des bras.
- **Une caméra au bout de chaque bras**, soit 3 caméras, juste à côté des têtes d'arrosage, permettant ainsi de diriger les bras directement sur les plantons.

Après quelques réflexions, il était clair que d'avantage d'informations mécaniques et de tests sur le terrain étaient nécessaires pour trancher la question. Afin de convenir à toutes les situations possibles, un maximum de 4 caméras à gérer a donc été définis, permettant ainsi d'utiliser les deux possibilités ci-dessus à la fois.

2.4.2 Spécifications

Un robot d'agriculture pouvant avoir un grand nombre de fonctionnalités différentes, il a fallu faire un choix quant à celles qui devaient être développées en priorité pour ce travail de bachelor. De plus, il fallait aussi s'assurer de la pérennité du projet ; il devrait être possible d'ajouter de nouvelles fonctionnalités le plus aisément possible. Les fonctionnalités qui ont été imaginées pour le robot ont donc été séparées en 3 catégories :

Fonctionnalités minimales : Il s'agit des fonctionnalités les plus importantes. Toutes ses fonctionnalités sont prévues dans le planning et devraient avoir été réalisées à la fin du temps alloué au travail de bachelor

Fonctionnalités souhaitables : Il s'agit de fonctionnalités relativement importantes. Certaines sont prévues au planning et il serait souhaitable de les réaliser pendant le travail de bachelor. Cependant, elles ne sont pas prioritaires et si le temps venait à manquer, pourrait ne pas être développées.

Fonctionnalités futurs : Il s'agit de fonctionnalités qui n'entrent pas dans le cahier des charges du travail de bachelor. Il s'agit là d'idées d'amélioration à apporter au projet par la suite. Dans la mesure du possible, le développement du robot doit être fait d'une manière à faciliter le développement et l'intégration de ces fonctionnalités.

A l'intérieur des 3 catégories, les fonctionnalités sont placées par ordre de priorité, les plus prioritaires venant en premier.

Fonctionnalités minimales

1. Déetecter correctement les plantons avec une ou plusieurs caméras
2. Commander un bras d'arrosage (ou tête d'arrosage) pour arroser les plantons détectés.
3. Commander la pompe ou de la vanne pour démarrer/arrêter l'arrosage
4. Déetecter le début d'une ligne dans un champ, se déplacer en suivant la ligne de terre qui peut être droite ou courbée en commandant les deux moteurs indépendants des roues et détecter la fin de la ligne.

Fonctionnalités souhaitables

1. Commander 3 bras d'arrosage (au lieu d'un seul)
2. Gérer les obstacles potentiels (les contourner ou s'arrêter si ça n'est pas possible)
3. Se déplacer de manière autonome (probablement par GPS) pour atteindre les lignes dans un champ depuis une station de recharge (eau, électricité) et vis versa
4. Arroser l'entier du champ en moins de 6h (dimensions du champs : voir page précédente)
5. Avoir une autonomie d'au moins 12h
6. Pouvoir recharger le robot, sans démonter les batterie, à l'aide d'un câble d'alimentation
7. Alerte par sms ou 3G (via app mobile) en cas de blocage, panne ou éloignement (vol))
8. Pouvoir refaire le plein d'eau de manière autonome
9. Étudier la possibilité d'utiliser un panneau solaire pour recharger
10. Développer une interface pour infos / réglages (écran ou web app ou client/serveur ou app mobile) :
 - Date / heure
 - Action en cours / programmées / terminées
 - Charge batterie / eau
 - Plan des lignes avec emplacement des plantons connus
 - Activer/Désactiver alertes

Fonctionnalités futures

1. Planter des graines ou des plantons
2. Déetecter les mauvaises herbes et pouvoir les gérer (écraser / couper / enlever)s.
3. Analyser l'humidité du sol et ajuster la quantité d'eau arrosée en conséquence
4. Créer une station de recharge solaire avec batterie où le robot pourrait venir se recharger de manière autonome

2.5 Et pour le reste...

2.5.1 Partie électronique

Entre la mécanique et l'informatique, se trouve l'électronique. Même si ce projet effectue plusieurs choix de type hardware, il y a tout de même une partie du robot qui n'est ni gérée par la partie mécanique, ni par la partie informatique. Il s'agit du choix de la batterie et de la manière de la charger. Il en va de même pour la réalisation d'un prototype. Même si ces quelques "détails" ne représentent pas une grosse quantité de travail, cette quantité est tout de même assez importante pour que ni la mécanique, ni l'information ne puisse s'engager à y consacrer du temps. Heureusement, d'autres collaborateurs au sein de l'école sont disponibles pour ce genre de tâche. Ils pourront s'en charger si aucun des deux étudiants ne trouve le temps de le faire.

2.5.2 Prototype

Avant de pouvoir adapter ce projet sur le produit final, c'est à dire, le robot développé par la partie mécanique, il a aussi fallu penser aux tests en laboratoire et au "Proof of concept". Une version prototype du robot ou du moins, des différentes entités mécaniques à commander devait être réalisée. Malheureusement, comme mentionné ci-dessus, cette réalisation n'entrait normalement dans aucun travaux de bachelor. La partie mécanique avait peut-être un petit robot qui allait peut-être pouvoir être adapté pour créer un prototype, mais rien n'était encore sûr. Cette décision a été remise à plus tard et comme nous le verrons plus tard dans ce rapport, c'est finalement dans le cadre de ce travail que le prototype a été conçu (voir chapitre 7).

Il a également été défini que le robot final (et éventuellement le prototype réalisé) devait pouvoir se piloter de manière manuelle (avec, par exemple, un joystick) et qu'un écran de 10' environ devait être embarqué sur le robot afin de pouvoir facilement obtenir des images et des informations en temps réels lors des tests. Cet écran devrait également être tactile afin de pouvoir paramétriser le robot facilement si besoin.

2.6 Étapes du projet

Une fois les fonctionnalités définies, il a fallu établir les différentes étapes du projet, et ce, de manière concrètes. Le projet a donc été décomposer en 3 objectifs principaux :

- Déetecter et arroser les plantons
- Avancer en suivant la bande de terre
- Se repérer et se déplacer dans le champ

La réalisation de ses trois objectifs se découpent en plusieurs étapes qui sont listées ci-dessous. Ces étapes ne distinguent pas les points minimaux ou souhaitables du cahier des charges. Elles ne seront donc pas toutes forcément terminées à la fin du temps alloué à ce travail de bachelor. De plus, chaque liste d'étape part du principe que nous partons de zéro et que les autres objectifs n'ont pas été réalisés. Certaines étapes sont donc redondantes à plusieurs objectifs. Cela permet une certaine flexibilité car il doit être possible de réaliser chaque objectif indépendamment des autres, et ce, dans n'importe quel ordre.

2.6.1 Déetecter et arroser les plantons

1. Choix du matériel (Carte et camera)
2. Commande du matériel
3. Réception du matériel, découverte et préparation pour le développement : Démarrage de la carte, installation des librairie, drivers et logiciel + Affichage image caméra
4. Acquisition de deux vidéos du champ avec les plantons à distances différentes
5. Développement soft reconnaissance plantons avec une caméra en «surplomb»
6. Tester le soft en marquant les plantons sur le flux vidéo
7. Réception d'un bras en plastique, de 3 moteurs et de la carte PWM, découverte et préparation pour les tests (avoir ou se créer un support pour le bras, qui le laisse libre de bouger)
8. Faire bouger les moteurs en ayant programmé les mouvements à l'avance
9. Faire bouger les moteurs de manière «manuelle», avec le clavier ou un joystick
10. Faire bouger les moteurs pour arriver à un point en coordonnées x/y/z (Traduction coordonnées ou offset x/y/z en coordonnées cylindrique (angles))
11. Coder la gestion automatique du bras et de l'arrosage
12. Simuler l'avance dans le champ et suivre les plantons avec soit :
 - Une grande feuille avec un bout du champ imprimé que l'on fait bouger sous le bras «à la main»
 - Une vidéo du champ filmée au paravent diffusée sous le bras (beamer ou grand écran)
 - Une longue feuille imprimée fixée sur deux rouleaux qui tournent qui défile sous le bras
13. Décider si une caméra supplémentaire au bout du bras est nécessaire. Coder le soft et tester si 2ème camera nécessaire
14. Faire des tests sur le terrain et des corrections une fois le robot disponible
15. Passer de 1 bras en plastique à 3 bras en acier et refaire les étapes 5 à 12.

2.6.2 Suivre la bande de terre

1. Choix du matériel (Carte et camera)
2. Commande du matériel
3. Réception du matériel, découverte et préparation pour le développement : Affichage image caméra, compilation, analyses électroniques, ...
4. Acquisition d'une vidéo du champ où les bords du champ sont visibles
5. Développement soft reconnaissance de la direction de la ligne choix de la direction à corriger (gauche ou droite)

6. Tester le soft en marquant les bords du champ sur le flux vidéo et en indiquant la correction de direction à effectuer
7. Simuler l'avance dans le champ et suivre la ligne avec soit : Une grande feuille avec un bout du champ imprimé que l'on fait bouger sous le bras «à la main» Une vidéo du champ filmée au paravent diffusée sous le bras (beamer ou grand écran) Une longue feuille imprimée fixée sur deux rouleaux qui tournent qui défile sous le bras
 - Une longue feuille imprimée qu'on fait bouger sous la caméra «à la main»
 - Une vidéo du champ filmée au paravent diffusée sous la camera (beamer ou grand écran)
 - Une longue feuille imprimée fixée sur deux rouleaux qui tournent qui défile sous le bras
8. Réception d'un petit robot à deux roues (fourni par Quentin Raball) dont les moteurs se commandent de la même manière que les moteurs finaux, découverte et préparation pour les tests.
9. Faire bouger le robot en ayant programmé les mouvements à l'avance (avance, recule, gauche, droite, plusieurs en même temps)
10. Faire bouger le robot de manière «manuelle», avec le clavier ou un joystick
11. Placer la camera sur le robot et simuler le suivi de la ligne en le faisant avancer sur une grande feuille avec une image du champ imprimée
12. Faire des tests sur le terrain et des corrections une fois le robot disponible

2.6.3 Se repérer et se déplacer dans le champ

1. Choix du matériel (Carte GPS (avec ou sans correction GSM) ou autre, peut-être camera supplémentaires ou capteurs)
2. Commande du matériel
3. Réception du matériel, découverte et préparation pour le développement : Affichage image caméra, réception donnée GPS, analyse électronique, ...
4. Développement soft choix de la direction à suivre en fonction de la position GPS (impossible de savoir dans quel direction est tourné le robot sans se déplacer un peu).
5. Bouger la carte avec le module GPS pour faire des tests de déplacement/distances/-directions/erreurs
6. Utiliser le petit robot de la partie «Suivre la ligne» pour simuler le déplacement dans le terrain, du réservoir d'eau au début d'une ligne.
7. Adapter ce soft pour qu'il démarre le programme «suivre la ligne» quand le points d'arrivée est proche (distance à définir)
8. Faire des tests sur le terrain et des corrections une fois le robot disponible
9. Développement du soft de la détection d'obstacle, éventuellement d'un passage entre les lignes au lieu d'aller tout droit.
10. Test avec le petit robot et des obstacles
11. Test sur le terrain et corrections

2.6.4 Planning

Ces différentes étapes ont pu être intégrées dans un planning qui se trouve en annexe de ce rapport. Au fur est à mesure de l'avance du projet et suite à quelques retards (voir

chapitre 8), ce planning a du être revu et ce, à deux reprises. Vous trouverez donc également 2 anciennes versions du planning. Le planning indique le nombre de jour prévu et le nombre de jour réellement effectué sur chaque étape. une valeur de 1 jour correspond à 8h de travail. Il est ainsi possible qu'il y ait plus de 5j effectués en une semaine.

Chapitre 3

Matériel

3.1 Besoins du projet

Pour pouvoir choisir et commander le matériel, il a d'abord fallu établir les besoins concrets et spécifiques du projets. Voici donc la liste concrète des actions que le montage final devra pouvoir entreprendre :

- Acquérir et traiter 4 flux d'images de manière simultanée
- Commander les 2 moteurs des deux roues indépendantes (2 PWM)
- Commander les 3 bras qui pourront avoir jusqu'à 3 moteurs chacun (9 PWM)
- Commander les vannes d'eau au bout de chaque bras (3 états logique)
- Configurer le robot à l'aide d'un écran tactile
- Diriger le robot de manière manuelle (à l'aide, par exemple, d'un joystick)
- Connaître la position GPS du robot et du bout des bras
- Établir une connexion TCP par GMS vers un serveur WEB ou une API distante.

Une fois les besoins définis, il a fallut procéder au choix du matériel. Le choix des composants principaux du projet sont décrits ci-dessous en 3 points :

- **Besoins** : Liste des spécifications nécessaires pour répondre aux besoin listés ci-dessus
- **Existant** : Description des différentes possibilités qui ont été analysées
- **Choix** : Synthèse et choix final.

3.2 Carte de système embarqué

3.2.1 Besoins

Pour des questions de simplification d'implémentation, le but est de n'avoir qu'un seul micro-contrôleur exécutant simultanément toutes les tâches du projet. Il est par contre tout à fait envisageable d'y ajouter d'autres cartes annexes qui contiendraient une partie hardware qui ne serait pas disponible sur la carte (puces, capteurs, ...). Cependant, il faudra prévoir la connectique nécessaire pour ces cartes annexes. D'autre part, vu les nombreux besoins du projet, un système d'exploitation semble être indispensable. Il permettra l'utilisation de nombreuses bibliothèques et simplifiera grandement l'implémentation de toutes les fonctionnalités. La puissance du micro-contrôleur devra donc être suffisante pour exécuter un OS, driver les périphériques annexes (caméras, GPIO, GPS, GSM, cartes annexes) et traiter 4 flux vidéos simultanément. Cette dernière tâche demandant largement plus de ressource que les autres, c'est elle qui définira la puissance minimum de la carte. Malheureusement, ne connaissant pas encore l'algorithme qui sera utilisé et vu le temps restreint disponible

pour le choix du matériel, il n'a pas été possible de définir précisément la puissance requise. Nous pouvons toutefois dire qu'un micro-contrôleur à 4 CPU sera très utile pour réaliser les 4 traitements de manière simultanée. De plus, la vitesse nécessaire dépendra fortement du nombre d'image à analyser dans le temps (image par seconde) et de leur résolution. Ces deux paramètres n'auront pas besoin d'être très élevés pour ce projet et pourront être diminués afin de réduire la puissance nécessaire des CPU ; 10 images/seconde en résolution standard devrait être largement suffisant. Dans la mesure du possible, il serait aussi préférable que la carte soit open-source.

Fort de ces 1ères analyses, voilà donc la liste des besoins de la carte embarquée :

- Processeur Quad-core
- Au minimum 1 port USB permettant de connecter un hub USB
- Interface permettant de dialoguer avec les éventuels périphériques externes (I2C, SPI, UART).
- Interface permettant la programmation, le debug et le transfert de fichier (UART ou JTAG ou Ethernet ou autre)

En outre, nous auront également besoin de la connectique et des périphériques suivants (pouvant être intégrés à la carte micro-contrôleur ou non) :

- jusqu'à 8 ports USB (4 caméras, tactile de l'écran, clavier éventuel, joystick de test, clé USB ou souris ou autre)
- Au moins 12 GPIO pouvant générer des PWM
- Puces GSM permettant une connexion 3G ou 4G
- Puces GPS précis à < 5cm
- 1 connexion pour un écran (probablement HDMI)

3.2.2 Existant

En regardant la grande liste des cartes embarquées recensées par wikipedia [3], on remarque que le choix est très vaste pour ce genre de projet. Néanmoins, voici 4 cartes intéressantes.

BeagleBone® Blue[4] Cette 1ère possibilité a été proposée par Jean-Pierre Miceli, un des ingénieurs de l'institut reds. L'avantage de cette carte est qu'elle a été conçue pour commander des moteurs de robotique et qu'elle inclus nativement beaucoup de sortie à cet effet. Malheureusement, il y a tout de même moins de 12 sorties PWM. Par contre, elle dispose d'un grand nombre de bus pour ajouter les périphériques annexes qui seraient nécessaires. Au niveau du micro-contrôleur, il s'agit d'un chip qui n'embarque une seul CPU (un Cortex-A8) tournant à 1GHz. Sa puissance risque donc d'être limite pour réussir à faire tourner l'algorithme de traitement d'image.

Raspberry PI 3B+[5] La marque raspberry PI est de loin la plus connues dans le domaine des petits ordinateurs embarqués. Elle jouit d'une communauté gigantesque et des centaines de projets et d'applications différentes ont déjà été réalisés à l'aide d'une de ces cartes. La sortie de la Raspberry PI 3 modèle B+, leur dernier modèle en date, a eu lieu en mars 2018, juste avant le début de ce travail de recherche. Cette nouvelle carte embarque un micro-contrôleur Broadcom BCM2837B0 comportant 4 CPU ARM Cortex-A53 tournant à 1,4GHz

et possède 1 GB de SDRAM LPDDR2. Au niveau de la connectique, elle possède 4 ports USB et bien qu'elle soit capable de générer des PWMs, elle ne possèdent pas suffisamment de sortie non plus pour driver tous les moteurs de ce projet. Elle dispose par contre de tous les bus et les interfaces nécessaires pour une ou plusieurs cartes d'extensions. Son arrivée récente sur le marché pourrait être source d'une fiabilité limitée. Cependant, en analysant les changements entre les deux versions, on s'aperçoit qu'ils ne sont pas très importants : Le micro-contrôleur Broadcom passe de la version BCM2837 à la version BCM2837B0, ce qui se traduit par une légère augmentation de la fréquence de fonctionnement (de 1,2 GHz à 1,4 GHz), la puce Wifi est maintenant Dual-Band au lieu de Single-Band, le protocole bluetooth passe de la version 4.1 à 4.2 et la carte peut maintenant être alimentée par le port Ethernet (Power over Ethernet). Aucune autre différence n'est décrite par le fabricant. Nous pouvons donc en conclure que ces quelques différences ne devrait pas influencer la fiabilité de la carte de manière significative. De plus, tous les projets et travaux disponibles sur internet, sont sans aucun doute compatibles avec cette version de la carte.

Banana PI M3[6] En matière de design hardware (taille, emplacement des connecteurs, vis, etc...), cette carte respecte la convention établie par Raspberry PI. La connectique est sensiblement la même que la raspberry 3B+ à l'exception de deux choses : elle possède un port SATA qui n'existe pas sur la raspberry mais n'est munie que de deux ports USBs. Par contre, au niveau de la puissance, cette carte est largement supérieur ! C'est d'ailleurs une des plus puissantes existantes dans le domaine des micro-ordinateurs embarqués. Le micro-contrôleur est un Allwinner A83T et possède 8 CPU ARM-A7 cadencés à 2GHz. De plus, la carte possède 2Gb de mémoire LPDDR3 avec un GPU plus puissant. Son prix est par contre doublé par rapport à la raspberry PI et la communauté d'utilisateur est plus réduite. De plus, elle semble avoir rencontré quelques problèmes de stabilité à sa sortie (cf article de castman.fr[7] en 2016). Il existe apparemment plusieurs versions de la carte, (utilisant notamment des connecteurs d'alimentation différents). Cette carte est donc une des plus puissantes sur le marché mais elle demandera quelques études supplémentaires pour être certain de ne pas avoir de problème avec.

Asus Tinker Board S[8] Cette carte fabriquée par ASUS pourrait être un très bon choix. En effet, c'est une carte très puissante : le micro-controlleur est un Rockchip RK3288 embarquant 4 CPU ARM Cortex-A17 tournant à 1,8GHz. De plus, elle possède un GPU à haute performance encore plus puissant que celui de la Banana PI qui pourrait être très bénéfique pour le traitement vidéo. Par contre, contrairement aux autres cartes présentées ici, elle n'est pas open-source. De plus, c'est une carte très récente, sortie en 2017 et pour ces deux raisons, la communauté utilisant cette carte est relativement restreinte. De plus, c'est la 1ère fois qu'Asus se lance dans les cartes embarquées de ce type et, malgré que la marque soit une référence en matière de fiabilité, la sortie est trop récente pour que la carte ait vraiment été testée et éprouvée par beaucoup d'utilisateurs. Les bibliothèques souvent utilisées sur d'autres support risquent donc de ne pas encore être optimisées pour fonctionner sur cette carte. Par exemple, d'après les quelques expériences d'utilisateurs de certains forums, il semblerait qu'il est difficile de faire fonctionner openCV (bibliothèque de traitement d'image utilisée pour le projet et présentée dans le prochain chapitre de ce document) sur le GPU au lieu du CPU. Cela engendrait donc une grosse perte de performance dans le cadre de notre projet. A noter toutefois que le design hardware de Raspberry PI est aussi respecté par cette carte.

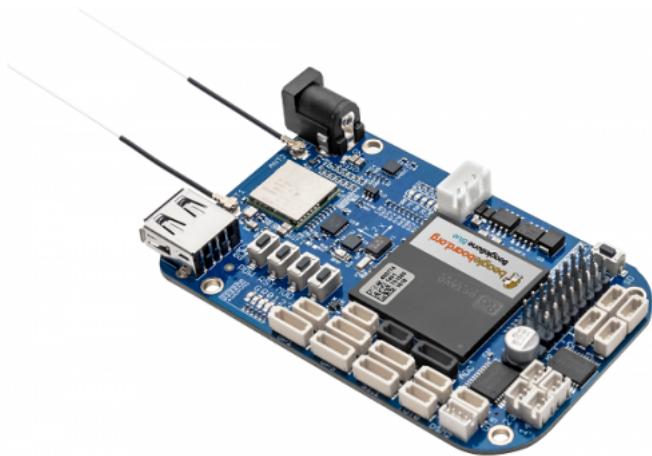


FIGURE 3.1 – BeagleBone Blue[9]



FIGURE 3.2 – Raspberry PI modèle 3B+[10]

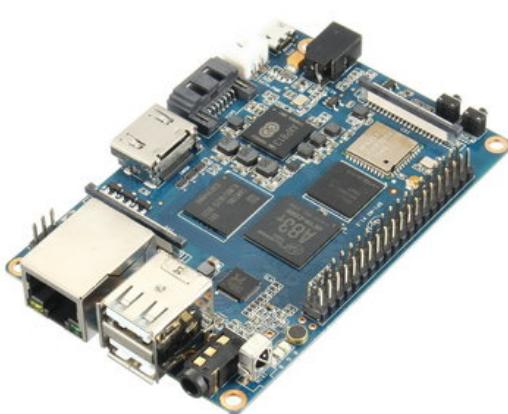


FIGURE 3.3 – Banana PI M3[11]

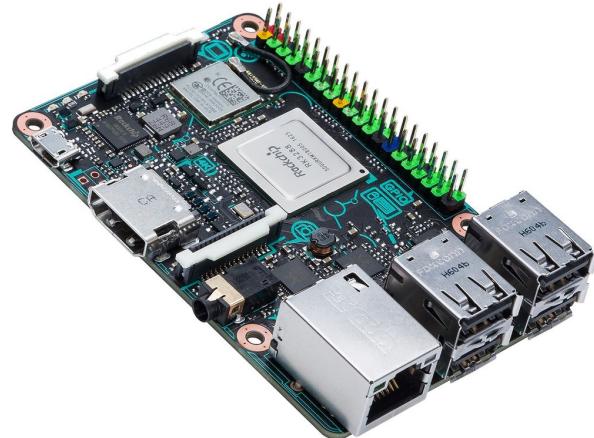


FIGURE 3.4 – Asus Tinker Board S[12]

Une comparaison détaillée de ces cartes a été réalisée à l'aide du site internet [board-tb\[13\]](#) et est disponible en annexe.

3.2.3 Choix

L'heureuse gagnante de cette sélection sera la **Raspberry PI modèle 3B+**. Au vue du peu de temps disponible pour ce projet, le plus simple semblait être le meilleur choix! Sa puissance devrait être suffisante et grâce à la communauté gigantesque de la raspberry PI, il est presque impossible que les problèmes que nous pourrions rencontrer n'aient pas déjà été traités et résolus par quelqu'un d'autre. De plus, les aides, les tutoriels et les projets utilisant une raspberry PI sont légions sur la toile et seront d'une aide précieuses. Et si vraiment un problème insoluble devait survenir, les utilisateurs des nombreux forums traitant de la raspberry sont actifs et nombreux et n'hésiteront pas à répondre aux demandes. Un support de qualité sera donc "naturellement" disponible.

Cependant, dans une future version du projet, ou si les performances de la raspberry PI se montraient insuffisantes, la banana PI ou la Tinker Board pourraient être d'excellents choix ! Ces dernières ne sont donc pas à ignorer.

3.2.4 Démarrage de la carte et installation d'openCV

La Raspberry PI est arrivée rapidement et le set choisi contenait une carte SD avec l'OS noobs[14] pré-installé. La sortie HDMI était activée par défaut et tout a avait l'air de fonctionner du premier coup. Après quelques jours d'études, il a fallu installer OpenCV sur la Raspberry PI (voir les raisons au chapitre suivant). La dernière version en date était la 4.0.0. Cette version s'est installée sans problème sur le PC de bureau mais, malheureusement, quelques soucis ont été rencontrés lors de la même opération sur la Raspberry PI (Erreur incompréhensible lors de la compilation des sources). Ces erreurs peuvent s'expliquer par le fait que la dernière version d'openCV est encore en développement et qu'elle n'est pas encore tout à fait stable. Finalement, c'est la version 3.2.0 qui a été installée sur la carte et tout fonctionne maintenant parfaitement. Par soucis de compatibilité, la même version a aussi été installée sur le PC (après avoir un bien sûr supprimer la version 4.0.0). Le choix de la version a été motivé par la trouvaille d'un tutoriel du site geckogeek[15] qui réalisait quelque chose de proche de notre objectif et qui utilisait cette version de la bibliothèque. Jusqu'à maintenant, tout s'est toujours bien passé avec cette version. Il n'a donc pas été nécessaire d'en changer. Toutefois, utiliser une version plus récente de cette bibliothèque peut s'avérer une bonne chose pour des questions de fonctionnalités, de performances et de support potentiel. Les détails d'installation sont disponibles en annexe, dans un fichier nommé "Installation et configuration"

3.3 Caméra

3.3.1 Besoins

En début de projet, les besoins concernant la caméra sont encore très flous. Nous ne connaissons pas la résolution nécessaire mais elle ne devrait pas être très élevée. Dans l'optique où la détection utilise les couleurs, le contraste et la luminosité de l'image devront par contre être les meilleurs possibles.

3.3.2 Existant

Webcam Logitech Pro C920[16] Cette webcam, bien que sortie en 2012, est aujourd'hui toujours classée dans les meilleures webcam du moment sur plusieurs comparatifs différents(voir par exemple : [17], [18] ou [19]). On devrait pouvoir donc avoir confiance dans sa qualité d'image. Néanmoins, elle ne fait pas partie des webcam les meilleurs marchés et devra utiliser un port USB de la carte. Pour les 1ers tests du projet, il n'est pas nécessaire de la commander mais elle reste une possibilité tout à fait valable pour une future version.

Raspberry PI Camera Module V2 et V2 Black Ces caméras possèdent l'avantage d'être complètement compatibles et optimisées pour fonctionner avec leur carte mère. Leur résolution maximum de 1080p est largement suffisante et ne sera probablement même pas utilisée. Leur petite taille les rendent très pratiques même si la place ne devrait pas être un problème sur le robot final. Elles se connectent via un port CSI dédié et libèrent ainsi un port USB. La

différence entre ces deux modèles est que la V2 Black a été conçu pour fonctionner dans des conditions de faibles éclairage et donne un bien meilleur rendu dans l'obscurité que la V2 classique. Par contre, les couleurs sont bien plus fades en luminosité correct. En parlant des couleurs, leur qualité ne sera certainement pas au niveau de la webcam ci-dessus mais celà devrait être suffisant pour détecter les plantons dans le champ. Par contre, la raspberry PI ne possédant qu'un seul port CSI, si le choix final est celui d'utiliser plusieurs caméras, il faudra, soit utiliser un autre modèle pour les cameras fixées au bout des bras, soit utiliser un adaptateur CSI -> USB. De plus, la nappe CSI d'origine étant limitée en longueur, il est aussi possible d'ajouter un câble HDMI avec les convertisseurs adaptés [20] pour pouvoir placer la caméra plus loin de la carte.

3.3.3 Choix

Le choix a donc été celui de commander les deux modèles des camera raspberry PI afin de pouvoir effectuer des tests et définir si leur qualité est suffisante pour le produit final.

Lors de la création du prototype pour les tests, une webcam d'un autre projet a été récupérée. Il s'agit de la **Logitech Quickcame Pro 9000**. Cette webcam n'est pas toute récente mais faisait partie des meilleurs du marché à sa sortie en 2007, elle fera parfaitement l'affaire pour donner une image à comparer à celles des deux autres caméras.



FIGURE 3.5 – Webcam Logitech Pro C920[21]



FIGURE 3.6 – Logitech Quickcame Pro 9000[22]



FIGURE 3.7 – Raspberry PI Camera Module V2[23]

FIGURE 3.8 – Raspberry PI Camera Module V2 Black[24]

3.3.4 Tests & Résultats

Une fois les 2 caméras commandées et réceptionnées, une petite série de test a permis de comparer le rendu des couleurs. Un test a été réalisé en pleine lumière, en laboratoire et un autre avec très peu de lumière. Chacun des deux tests a été réalisé avec 2 résolutions différentes. Les images de la webcam Pro 9000 ont aussi été ajoutée à titre de comparaison. Le format d'image de la webcam est pas contre en 4 :3 contrairement à celui de la camera de raspberry qui est en 16 :9. La résolution n'est donc pas exactement la même mais les ordres de grandeur sont respectés.

Test 1 - Lumières allumées

Résolution : 640x480



FIGURE 3.9 – Raspberry Camera Module V2



FIGURE 3.10 – Raspberry Camera Module V2 black



FIGURE 3.11 – Webcam HD Pro 9700

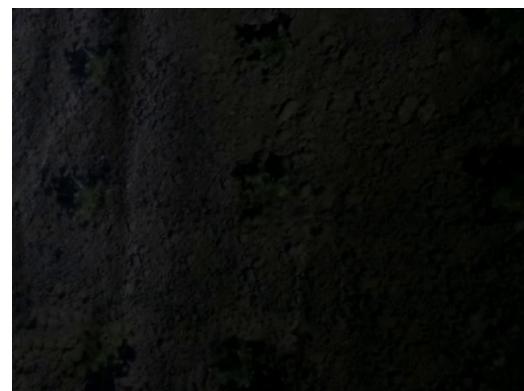
Résolution : 1280x720



FIGURE 3.12 – Raspberry Camera Module V2



FIGURE 3.13 – Raspberry Camera Module V2 black

Test 2 - Lumière très faible**Résolution : 640x480****FIGURE 3.14 – Raspberry Camera Module V2****FIGURE 3.15 – Raspberry Camera Module V2 black****FIGURE 3.16 – Webcam HD Pro 9700****Résolution : 1280x720****FIGURE 3.17 – Raspberry Camera Module V2****FIGURE 3.18 – Raspberry Camera Module V2 black**

Analyse des résultats et conclusion

Ces tests montrent très clairement que les deux cameras de la Raspberry PI sont bien en dessous d'une webcam traditionnel. De plus, la version black qui est sensé mieux fonctionner dans la nuit, fonctionne en réalité presque moins bien que la version classique. Ces caméras n'ont toutefois pas été testées en conditions réelles, avec la lumière du soleil. Les résultats seront peut-être meilleurs. Cependant, pour le projet final, il faudra tout de même très probablement commander une véritable webcam (la C920 par exemple). En attendant, la Pro 9000 utilisée ici sera suffisante pour les premiers tests.

3.4 Écran

3.4.1 Besoins

Concrètement, il n'est pas certains du tout que l'idée d'un écran soit conservée sur la version finale du robot. En effet, l'environnement extérieur au robot lors de son utilisation finale peut être dommageable pour ce matériel. Il faudrait donc prévoir une coque transport ou n'importe quelle autre moyen permettant de le protéger des choses, des intempéries, de la poussières, etc... De plus, il pourrait être remplacé par une interface WEB ou mobile extérieur au robot. Et même si un écran était finalement nécessaire, un petit écran LCD monochrome serait suffisant pour indiquer et configurer les quelques paramètres nécessaires. L'écran décrit ici risque donc d'être utilisé uniquement lors des phases de tests et pour la 1ère version du robot final. Voici une liste des "besoins" pour cet écran :

- connecteur hdmi
- Taille : entre 8' et 12' environ
- Tactile appréciable
- Compatible avec Raspberry PI

3.4.2 Existant

Il existe beaucoup d'écrans différents compatibles avec la raspberry pi, voir même, conçus spécialement pour ce genre de board. Pour les départager, en plus des besoins décrits ci-dessus, les critères suivants ont été pris en compte :

- Résolution et qualité de l'image
- Facilité de gestion du tactile (driver nécessaire ou non)
- Prix à l'achat
- Recommandations et commentaires des utilisateurs

En plus de l'écran finalement choisi décrit ci-dessous, voici 2 autres alternatives :

Elecrow 11.6'[25] : L'avantage de cet écran est qu'il semble bien robuste. De plus, sa résolution de 1920x1080 est très importante pour sa taille et la qualité d'image n'est pas en reste. Malheureusement, puisqu'il n'est pas tactile et que son prix de 139\$ est important (presque aussi chère d'un écran de PC), ce n'est pas ce modèle qui a été choisi. Il reste cependant un produit intéressant.

Banana PI tactile 9'[26] : Plus petit et beaucoup moins cher que l'écran ci-dessus, voici un écran produit par Banana PI et spécialement conçu pour fonctionner avec des board tel que la raspberry PI ou la Banana PI. Il est tactile et aucun driver n'est nécessaire pour le faire fonctionner, le tactile est reconnu comme une souris. Par contre, il nécessite des précautions particulières pour le monter et sa résolution n'est pas incroyable (seulement 1024 x 600 pixels). De plus, l'alimentation (entre 9V et 12V) n'est pas fournie. Ces 3 points négatifs en font au mauvais candidat pour un produit de prototypage mais il peut s'avérer une bonne alternative s'il finalement décidé d'un écran tactile doit être intégrer au robot final.

3.4.3 Choix

Le choix s'est finalement porté sur **un ecran de chez Waveshare[27]**. Actuellement, il n'est pas encore certain qu'un écran tactile soit indispensable. En effet, il est tout à fait possible de commander le robot par un api, en 3G, à distance, ou en se connectant en wifi avec un smartphone et en utilisant une API dédiée. Il est également possible d'ajouter au robot quelques boutons pour le paramétrier. Cependant, pour tous les tests, pour des questions de flexibilité, une version tactile de l'écran semblait être le plus approprié. Si par la suite, il s'avère que la propriété tactile est effectivement inutile mais que l'écran est nécessaire, il sera toujours possible de commander un autre écran, moins cher.



FIGURE 3.19 – Écran Elecrow 11,6'[28]



FIGURE 3.20 – Écran tactile Waveshare 10,1'[29]

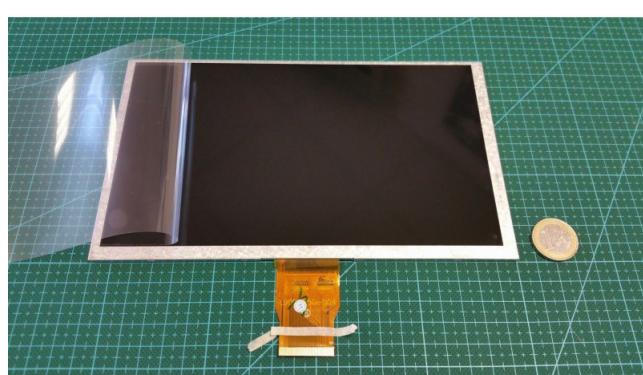


FIGURE 3.21 – Écran tactile Banana PI 9'[30]

3.4.4 Test du Waveshare 10,1'

Le rendu des couleurs est très correct. Sur Raspbian (l'OS de la Raspberry PI), le tactile a fonctionné du 1er coup, sans qu'aucune installation n'ai été nécessaires. Il a par contre été nécessaire de télécharger une clavier visuel pour pouvoir être totalement indépendant avec cet écran.

3.5 Carte GPS

3.5.1 Besoins

Le positionnement par GPS sera utilisé pour deux fonctionnalités :

- Permettre au robot de se diriger dans le terrain, pour retrouver sa station de base et le début des lignes
- Enregistrer et reconnaître la position des plantons au bout des bras d'arrosage

Pour la 1ère utilisation, le robot pouvant s'aider de sa caméra, une puce GPS de précision standard (entre 1m et 2m) devrait être suffisante. Par contre, pour cette 2ème utilisation, vu la petite taille des plantons et leur faible espacement par rapport au potentielles mauvaises herbes, nous avons besoin d'un GPS très précis (de l'ordre de quelques cm seulement).

3.5.2 Existant

L'erreur standard des GPS (quelques mètres seulement) est dûe aux déviations/modifications par la couche d'ozone, des signaux utilisés pour communiquer avec les satellites. Il existe plusieurs technologie de GPS différentes et certaines réduisent cette erreur. Sans entrer dans les détails, si plusieurs appareils GPS sont au même endroit, au même moment, leur erreur (déviation) est la même pour les différents gps. Ainsi, il existe des bornes GPS fixes dont la position exacte (sans erreur) est connue. Ces bornes reçoivent un signal GPS et peuvent donc connaître l'erreur actuel d'un lieu. Elles peuvent envoyer via GSM la correction de cette erreur pour permettre aux autres appareils GPS abonnés dans la zone de se recalibrer. Cette technologie permet une précision de l'ordre de quelques centimètres seulement. Pour plus d'informations sur les technologiques GPS existante, quelques pages expliquant la technologie GPS[31] et les causes des erreurs sont annexées à ce rapport.

Il est par contre très difficile de trouver sur le marché des cartes seule à une puce GPS sans qu'elles fasse partie d'un produit finis. Un modèle a toutefois été trouvé. Il s'agit d'une petite carte de réception GPS de chez Embedded Artist (Figure 3.22).

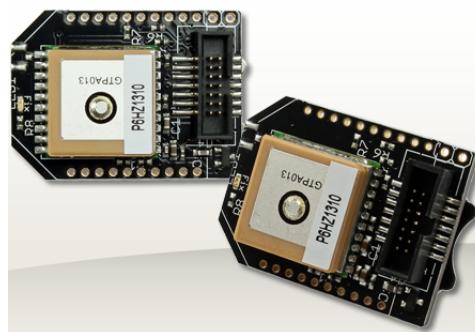


FIGURE 3.22 – GPS RECEIVER BOARD[32]

3.5.3 Choix

La Heig-VD entretient de bonnes relations avec une entreprise de puces GPS nommée **U-blox**. Cette entreprise fournit le genre de puces dont nous aurions besoin mais seulement en grande quantité. Nous les avons contacté pour savoir s'il était possible de faire une exception et n'avons pas encore reçu de réponse à ce jour. Pour finir, l'idée de mettre une puce GPS sur le robot a été abandonnée pour sa première version mais reste une nécessité pour une future évolution du projet.

3.6 Carte GSM

3.6.1 Besoins

Comme expliqué dans cette article [33] du site aruco, la carte SIM est encore aujourd'hui nécessaire pour assurer une connexion en extérieur. Notre robot devra donc être muni d'une puce GSM et d'une carte SIM afin d'assurer une connexion à internet pour la liaison TCP de la futur-interface.

3.6.2 Existant

Ayant finalement mis la priorité sur d'autres points du développement, la question n'a été que peu analysée. Toutefois, il est intéressant de noter qu'il existe des cartes SIM multi-opérateur à coût très réduits (voir [34]). Il existe aussi des modules avec carte GMS compatible avec la raspberry PI (par exemple le module SIM9000 ci-dessous)

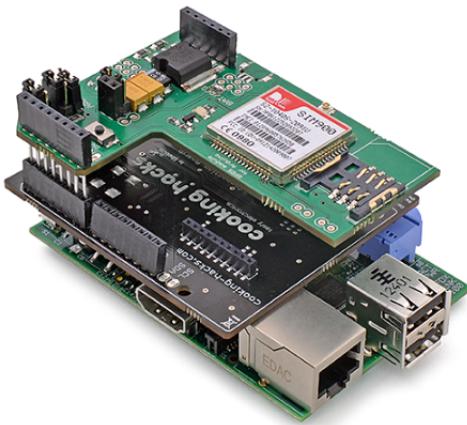


FIGURE 3.23 – GPRS/GSM Quaband Module for Arduino and Raspberry Pi Tutorial (SIM900)[35]

3.6.3 Choix

Puisque la connexion GMS ne sera pour finir par utilisée pour cette 1ère étape projet, le choix réel d'une carte à puce n'a pas été terminé et se fera dans une future étude de la suite du projet.

3.7 Carte d'extension PWM

3.7.1 Besoins

Étant donné que la Raspberry PI ne possède pas suffisamment de sortie PWMs nativement, il est nécessaire d'ajouter au système une carte d'extension. Cette dernière doit posséder au moins 12 sorties PWM et doit pouvoir être dirigée par la raspberry PI (I2C, SPI ou UART).

3.7.2 Choix

Le carte choisie est **une carte d'extension de chez Adafruit**. Cette dernière se commande par le bus I2C et comporte 16 sorties PWMs. De plus, si ce nombre venait à être insuffisant pour le futur du projet, il est à noter que 6 bits de l'adresse I2C de ces modules sont paramétrables. Il est ainsi possible de monter 64 de ces cartes en parallèle et de toutes les diriger par le même bus I2C. Attention toutefois : la fréquence des PWMs n'est pas paramétrable. Elle est fixé à 200Hz ce qui est compatible avec les moteurs de modélisme du prototype. Son data sheet est donné en annexe et la description de son fonctionnement est donné en même temps que les tests des moteurs au chapitre 5. Etant donné que les moteurs du robot final ne seront pas piloté par PWM (voir chapitre 5), il faudra, dans la suite du projet, une fois les moteurs finaux connus, rechercher des drivers hardware compatibles.

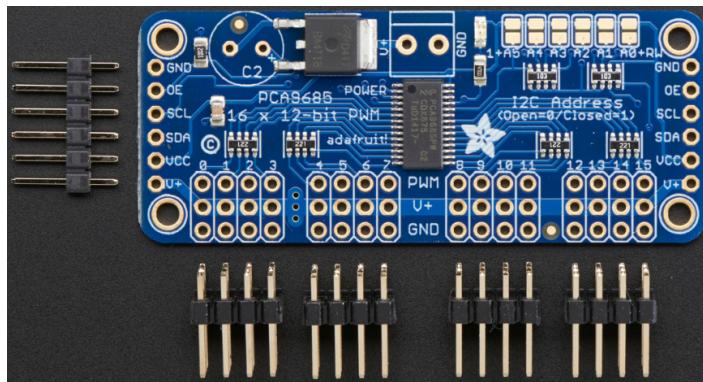


FIGURE 3.24 – Carte d'extension PWM de chez Adafruit[36]

3.8 Synthèse : Ce qu'il faut encore faire au niveau hardware

Voici un résumé des différentes tâches qu'il est nécessaire de faire au niveau du matériel (classées par ordre de priorité) :

1. Se procurer un moteur pas à pas et un driver au plus proche des produit du robot final et adapter le soft pour les faire fonctionner
2. Recontacter la société U-blox (essayer peut-être via le forum cette fois), pour leur demander une solution GPS convenable pour le projet ou se diriger vers une autre entreprise.
3. Se pencher plus précisément sur la question de la puce GMS et voir s'il est possible d'utiliser celle qui servirait à corriger l'erreur du GPS
4. Tester la qualité des caméras en extérieur et se décider à passer à une webcam de meilleure qualité
5. Dans le cas où les performances de la raspberry PI se montraient insuffisantes sur le terrain, il faudrait commander et porter le projet sur une board plus puissante

Chapitre 4

Reconnaissance des plantons

4.1 Les différentes possibilités

Pour pouvoir diriger les bras d'arrosage sur les plantons, la 1ère chose à faire est de connaître leur emplacements. Il faudra donc les reconnaître sur l'image de la (des) caméra(s). Il existe plusieurs moyens pour cela :

La détection des contours et des formes : Cette méthode existe mais, dans le cas présent, elle serait très difficile à faire fonctionner. En effet, la forme des plantons est beaucoup trop variable pour que cette méthode soit efficace. Cette possibilité a donc directement été écartée.

La différentiation par les couleurs : C'est une méthode relativement facile à implémenter qui n'est pas trop gourmande en ressource. Elle a par contre plusieurs désavantages. Tout d'abord, tous les plantons n'ont pas la même couleur. Il faudra donc prévoir plusieurs versions de l'algorithme en fonction du type de planton sur la ligne (exemple : les salades rouges). Ensuite, elle est loin de fonctionner parfaitement dans tous les cas et risque de souffrir des facteurs extérieurs tel que les changements de luminosité. Enfin, la présence d'autres objets de la même couleur mais qu'il ne faudrait pas arroser (comme des mauvaises herbes par exemple) serait assez difficile à gérer. Néanmoins, en paramétrant correctement la couleur et la tolérance acceptée, ainsi qu'en effectuant quelques opérations d'analyse d'image, il devrait être possible d'arriver à un résultat acceptable. C'est donc cette méthode qui a été choisie, pour une 1ère version en tout cas.

Les réseaux de neurones : L'intelligence artificielle étant en plein essor, cette méthode semble être la plus efficace pour réaliser la reconnaissance d'objet. Le problème est que les réseaux de neurones demandent une grande quantité de données sources pour les phases d'apprentissage et de validation. Nous ne disposons pas de ses données et les acquérir ou les créer demanderaient un temps de travail très important. De plus, il serait aussi fastidieux de créer un réseau de neurones adapté et optimisé pour ce cas. Étant donné le temps restreint alloué à ce travail de bachelier et le fait que l'objectif de ce projet est surtout d'avoir un robot fonctionnel en entier en fin de semestre et non uniquement la reconnaissance des plantons, nous n'utiliserons pas de réseau de neurone pour cette 1ère version du robot. Cela reste toutefois une option valable et pourrait être une excellent alternative pour les futures versions du projet.

4.2 Plantons ou mauvaises herbes ?

La question de la différentiation des plantons et des mauvaises herbes est importante. Avant de parler plus en détail de l'algorithme de reconnaissance des couleurs, il semblait important de trouver un ou des moyens de paliers à ce problème. En effet, la couleur des plantons et des mauvaises herbes risquent d'être sensiblement la même. Comme éviter d'arroser aussi les mauvaises herbes ?

4.2.1 Position GPS

Si le GPS s'avère suffisamment précis, une idée serait de sauvegarder la position des plantons juste après que ceux-ci aient été plantés. A ce moment, la terre viendrait d'être retournée et il n'y aurait donc pas de mauvaises herbes présentes. Cette solution aurait l'avantage de pouvoir ensuite détecter la présence de mauvaises herbes pour pouvoir les signaler voir même, les traiter avec une future version du robot.

4.2.2 Affiner les paramètres de détection

Dans le cas où le GPS ne serait pas utilisable, il est aussi possible de jouer sur les paramètres de l'algorithme de reconnaissance. La taille et la couleur des mauvaises herbes étant rarement exactement les mêmes que celles des plantons, il devrait être possible de diminuer les faux positifs, mais cette méthode ne pourra peut-être pas supprimer complètement le problème. C'est donc une question qui reste en suspend et qui se devra d'être traitée à posteriori.

4.3 Bibliothèque choisie

Pour tout ce qui est du traitement d'image, la bibliothèque OpenCV est une vraie référence. Elle est grandement utilisée et il existe sur internet plusieurs tutoriels décrivant son utilisation, ainsi que beaucoup d'exemples de projets effectués avec ses fonctionnalités. Son installation sur la Raspberry PI a aussi été effectuées par de nombreuses personnes. En outre, cette bibliothèque est open-source et existe depuis de nombreuses années. Le code n'en est donc pas à sa 1ère version et a déjà été testé et optimisé à de nombreuses reprises. Et pour finir, c'est une bibliothèque qui a déjà été utilisée par plusieurs personnes dans l'institut reds. En cas de problème, une aide ne serait donc pas difficile à obtenir.

4.4 Algorithme

Une grosse partie de ce travail de bachelor a été d'élaborer un algorithme suffisamment efficace et suffisamment optimisé pour pouvoir détecter correctement les plantons sur un maximum d'images différentes. Cet algorithme ne prétend pas être le plus optimisé ou le plus efficace mais il est le fruit de plusieurs réflexions et a fait l'objet de plusieurs ajouts et modifications tout au long du projet, notamment suite aux différents tests effectués. Un tutoriel du site geckogeek[15] a été particulièrement utile pour comprendre et poser les bases de la reconnaissance d'objet par la couleur avec opencv. Ce tutoriel explique comment reconnaître un objet d'une couleur donnée dans un flux vidéo provenant d'une webcam. La base du présent algorithme a été inspirée de ce tutoriel mais plusieurs modifications ont

été apportées afin de le perfectionner au mieux pour le cas précis des plantons en terre. L'algorithme se déroule en 5 étapes :

1. Binarisation
2. Opérations morphologiques
3. Séparations des objets (en ignorant les objets trop petits)
4. Calcul des positions
5. Suppression des objets trop proches les uns des autres

Chacune de ces étapes est décrites en détail ci-dessous. Pour un exemple visuel, il suffit de regarder plus loin dans ce chapitre : les résultats des tests montrent le rendu de l'image après chacune des étapes.

4.4.1 Binarisation

La binarisation, au sens large du terme, consiste à séparer les pixels d'une image en 2 catégories distinctes. Dans notre cas, il s'agit de créer une image de la même taille que l'image source mais avec uniquement des pixels noirs et des pixels blancs. En blancs, les pixels de la couleur des plantons (un vert légèrement jaunâtre, dans une certaine tolérance) et, en noir, tous les autres pixels. La figure 4.1 ci-dessous montre un exemple de binarisation sur une petite image de 4x4 pixels agrandie.

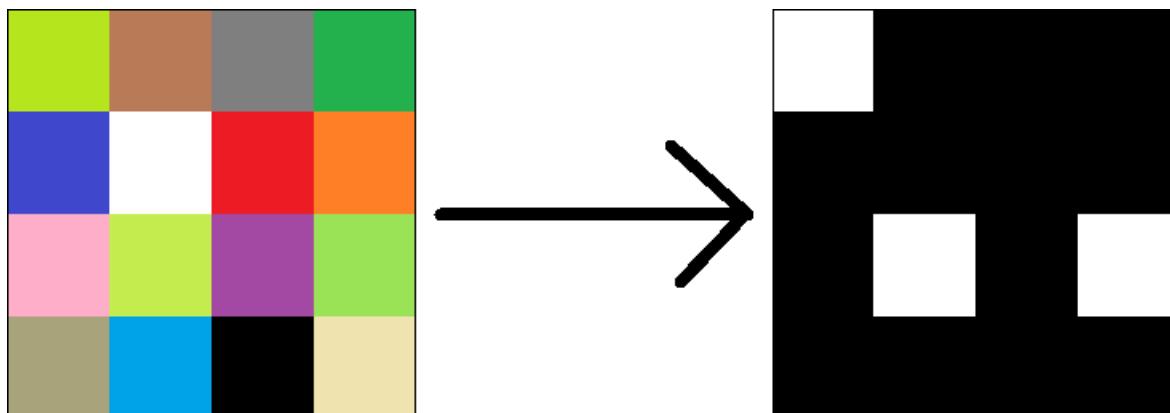


FIGURE 4.1 – Exemple d'une binarisation sur une image de 4x4 pixels

Format de la couleur

Il existe plusieurs formats différents utilisés en informatique pour exprimer une couleur. De manière générale, le format le plus utilisé est le format RGB (Red, Green, Blue en anglais). Il s'agit d'un niveau variable de rouge, de vert et de jaune. Ce format est par exemple utilisé dans les fichiers bitmap (bmp), qui correspondent au format d'image le plus simple (non compressé). Pour ce projet, les images sources utilisées, provenant de la webcam, sont au format RGB. Toutefois, en traitement d'image, on préfère utiliser le format HSV. Les lettres HSV signifient Hue Saturation et Value, soit Teinte, Saturation et Valeur en français. Dans notre cas, chaque valeur est codée sur 8 bits et a donc une plage de 256 valeurs différentes (de 0 à 255), ce qui fait au total $256^3 = 16,8Mio$ de couleurs possibles. Voici la description des ces 3 valeurs[37].

Teinte : Cette valeur représente la couleur du pixel. Cette valeur est proportionnelle à l'angle de la position de la couleur sur le cercle chromatique (Figure 4.2). Avec une règle de trois, il facile de transformer l'échelle en angle (sur 360) ci-dessus en valeur sur 8 bits (sur 256). Par exemple, le bleu ayant un angle de 180° sur ce cercle se traduit par la valeur $\frac{255}{360} \cdot 180 = 128$

Saturation : Cette valeur correspond à la quantité (ou l'intensité) de la teinte. Plus cette valeur est élevée, plus la couleur est vive. A l'inverse, une valeur de 0 correspond à un pixel gris.

Valeur : Aussi appelé "Brillance", cette valeur représente le niveau de lumière de la couleur. Plus cette valeur est élevée, plus le pixel se rapproche du blanc. A l'inverse, un 0 correspond à un pixel noir.

Si ce format est utilisé en traitement d'image, c'est en partie parce qu'une des valeurs représente uniquement la luminosité (ce qui n'est pas le cas du format RGB par exemple). Il est ainsi possible (et c'est ce que fait cette algorithme) d'ignorer ce paramètre. On réduit ainsi en partie les problèmes de détection probables dû à la luminosité de l'image (météo, heure de la journée, ombres diverses, ...) en tenant compte uniquement de la teinte et de la saturation. A noter que le format YUV (aussi appelé YCbCr) aurait été aussi une possibilité puisque la valeur Y (Luminance) correspond aussi à la luminosité d'un pixel (voir la dernière section de ce chapitre). Le format HSV a toutefois été préféré car il donnait de meilleur résultat lors de la binarisation grâce à la saturation qui permet d'inclure beaucoup plus de pixel dont la couleur varie autour de la même teinte (voir ci-dessous).

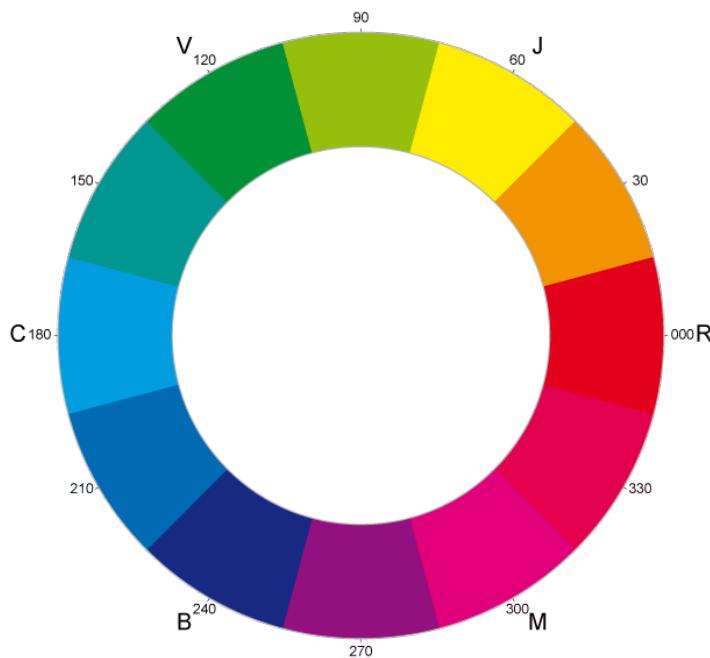


FIGURE 4.2 – Cercle chromatique[38]

Couleur verte et tolérances

Pour définir ensuite les valeurs HSV correspondante ou pas à un plantron, les images d'une vidéo du champ ont été utilisées. Grâce à une option du programme réalisé dans le cadre de ce travail permettant de récupérer la valeur de la couleur du pixel cliqué par la souris et de l'enregistrer dans un fichier, il a été possible de récupérer un tableau avec environ 150 échantillons de valeurs HSV correspondant à un même type de plantron. Ce

fichier s'est finalement avéré surtout utile pour calculer la valeur H (teinte) de la couleur à détecter : La moyenne arithmétique définit la valeur de base et l'écart type nous donne la tolérance à utiliser. Le tableau contenant les valeurs HSV utilisées ainsi que les calculs statistiques est disponible en annexe. Pour la valeur S (Saturation), les tests ont montré qu'il fallait mieux utiliser une large plage de cette valeur. Concrètement, seule une limite inférieure est finalement été utilisée. Cela rend la binarisation moins propre (avec plus de bruit), mais renforce davantage les zones de plancton. Les opérations morphologiques sont ainsi plus efficaces et le rendu final est bien meilleur. En ce qui concerne la valeur V (luminosité), comme expliqué ci-dessus, elle n'est pas prise en compte (-> plage acceptée de 0 à 255).

4.4.2 Opérations morphologiques

A ce stade, bien que l'ont devine l'emplacement des plantons, il reste encore beaucoup de bruit et de faux positif sur l'image. Pour corriger cela, nous faisons appel à 2 opérations morphologiques : L'érosion et la dilatation. Les opérations morphologiques sont des filtres qui modifient une image en fonction d'un élément d'une forme spécifique appelé "élément structurant" ou "noyau". L'érosion supprime les pixels isolés (ou les petits groupes) tandis que la dilatation renforce les amas de pixel. Plus l'élément structurant est grand, plus l'opération morphologique sera importante (mais plus elle prendra du temps). Une érosion suivie d'une dilatation s'appelle une ouverture. L'inverse (une dilatation puis une érosion) s'appelle une fermeture. Pour plus de détail, il ne faut pas hésiter à aller faire un tour sur la page internet de Christian RONSE [39] qui explique très bien ces opérations.

Plusieurs séries d'opérations différentes ont été testées pour trouver celles donnant le meilleur résultat. Ainsi, pour faire ressortir uniquement les plantons de l'image, dans tous les cas testés lors de ce travail, c'est une ouverture seule (érosion puis dilatation) qui donne le meilleur résultat. Toutefois, ayant testé différentes séries d'opération, le programme actuel permet également de faire suivre l'ouverture par une fermeture (dilatation puis érosion). Cette possibilité a été conservée pour une potentielle utilisation future. Il suffit de donner un noyau de taille 0 à la fermeture pour qu'elle ne soit pas appliquée. C'est ce que fait actuellement le programme par défaut.

4.4.3 Séparation des différents plantons

L'image montre maintenant les plantons sous la forme de masses de pixels bien distinctes. Il faut donc calculer l'emplacement de chaque planton et pour ça, il est nécessaire de savoir quel pixel appartient à quel plancton. Pour ce faire, deux fonctions ont été codées : splitObjects et analyzePixel. analyzePixel est une fonction récursive appelée par splitObjects. L'idée est de créer une matrice de la même taille que l'image. Chaque valeur de cette matrice correspond à un pixel de l'image. L'algorithme va remplir cette matrice avec des valeurs correspondant aux IDs des objets auquel appartiennent les pixels. Au début, tous les pixels ont un ID de 255, qui indique que le pixel n'a pas encore été analysé. La fonction splitObjects va appeler analysePixel sur tous les pixels de l'image, en commençant par le pixel de coordonnée (0;0) (en haut à gauche) puis ligne par ligne, de gauche à droite et de haut en bas. Lorsque la fonction analysePixel rencontre un pixel noir avec l'ID 255 (non analysé), il lui donne l'ID 254, ce qui indique que le pixel a été analysé mais ne correspond à aucun objet. La fonction se termine alors en retournant la valeur 0. Lorsque qu'un pixel blanc avec l'ID 255 (non analysé) est rencontré, analysePixel lui donne l'ID que splitObjects lui aura

passé en paramètre puis se rappelle elle-même sur les quatre pixels adjacents (il y a donc récursion). Lorsque la récursion est terminée, analysePixel renvoie, dans ce cas, la valeur 1. Si un pixel déjà analysé est rencontré (ID différent de 255), la fonction retourne 0 directement. Une pixel ne sera donc jamais analysé plusieurs fois. Attention : Si les amas de pixel sont trop importants, la limite de récursion est atteinte. C'est le cas uniquement si les opérations précédentes de l'algorithme n'ont pas été effectuées correctement. Si cette limite est atteinte, la détection en cours est annulée afin d'éviter un plantage du logiciel. La valeur de retour de analysePixel est utilisée uniquement par splitObjet pour savoir si oui ou non, un nouvel objet a été détecté. Si c'est le cas, alors splitObjects va décider si la taille de l'objet est suffisante en regardant le nombre de pixel de ce dernier. Si c'est le cas, on rappelle analysePixel sur le prochain pixel avec un nouvel ID. Si ce n'est pas le cas, on modifie l'ID de tous les pixels de cet objet trop petit en 254 (déjà analysé mais n'appartenant à aucun objet) puis on rappelle analysePixel sur le pixel suivant sans changer d'ID. De cette façon, tous les pixels blancs qui se touchent auront le même ID et appartiendront donc à un même objet et les derniers amas de pixels que les opérations morphologiques n'auront pas réussis à éliminer ne seront donc pas considérés car ils seront trop petits. Voici un résumé de la signification des IDs des pixels :

255 : Non analysé

254 : Analysé mais ne fais pas partie d'un objet

0 - 253 : ID correspondant à un objet suffisamment grand

Gestion de la perspective

Suivant le design mécanique du robot, il est possible que la vision de la caméra ne soit pas vraiment perpendiculaire au plan du champ. Ceci peut induire une notion de perspective assez importante pour qu'il soit nécessaire de la prendre en compte pour définir la taille minimum d'un objet sur l'image. Ainsi, la comparaison de la taille tient compte de la position de l'objet sur l'image. Plus il est en haut de l'image (c'est à dire loin du robot en réalité), plus il peut être petit. À l'inverse, plus il est en bas de l'image (c'est à dire proche du robot en réalité), plus il doit être gros pour être détecté. Le rapport entre la taille des objets lointains et des objets proches est paramétrable car il dépend de l'angle de la caméra.

4.4.4 Calcul des positions et élimination des objets trop rapprochés

L'algorithme ci-dessus a aussi additionné toutes les coordonnées X et toutes les coordonnées Y des pixels d'un même objet. En divisant ainsi ces deux sommes par le nombre de pixel de l'objet (qui a aussi été sauvegardé), on obtient le centre de gravité (barycentre) de l'objet. Une fois cette valeur connue et grâce à Pythagore, nous pouvons calculer la distance (en pixel) d'un objet à un autre. Toutes les distances sont vérifiées et si deux objets sont trop proches l'un de l'autre, le plus petit des deux (toujours en nombre de pixel) est supprimé. Une 2ème idée était "d'additionner" les deux objets, c'est à dire, donner le même ID à chacun des pixels des deux objets. La position de ce nouvel objet ainsi créé serait donc le centre des deux anciens objets proches. Il s'avère que, lors des tests, puisque l'opération morphologique fonctionnait très bien, le 2ème petit objet proche du 1er plus grand appartenait presque toujours à autre chose qu'à un plantron. Cette 2ème idée a donc été écartée. Il est tout de même bon de la garder en tête pour la suite du projet. La notion de perspective est ici aussi importante et se gère de la même manière que ci-dessus pour la taille de l'objet : Plus

les objets sont éloignés, plus la distance minimum entre eux est petite. Plus ils sont proches, plus cette distante est importante.

4.5 Paramètres, options et optimisations

Le logiciel de reconnaissance a été codé de manière à pouvoir être utilisé dans un maximum de situations différentes. Dans cette idée, beaucoup de paramètres et d'options ont été implémentés. Quelques options nécessitent une modification de constantes dans le code et une recompilation, la plupart peuvent être modifiées au démarrage du programme (en ligne de commande) et certaines peuvent même être modifiées de manière dynamique, pendant l'exécution du programme. Tous ces paramètres ont été très utiles pendant les phases de test car ils ont permis d'effectuer plusieurs tests différents sans devoir recompiler le logiciel voir même, sans devoir redémarrer l'application. Le temps alloué à la réalisation de ses différentes options a été plus important que prévu car elles n'avaient tout simplement pas été pensées à la base. Néanmoins, elles se sont révélées très utiles lors de l'étude de l'algorithme et des tests. De plus, elles seront presque indispensables au moment de l'intégration et de l'adaptation du projet au matériel final (robot et caméra) puisque la détection des plantons dépend grandement du matériel utilisé et de l'environnement. Ces options seront aussi utiles pour pouvoir ajouter facilement différents types de légumes à arroser. Un manuel de l'utilisateur (disponible en annexe) a également été écrit et décrit leur utilisation plus en détail. Voici les raisons qui conduisent à l'implémentation des options concernant le traitement d'image.

4.5.1 Source, fenêtres de sortie, résolution et taille d'affichage

Tout d'abord, pour pouvoir effectuer des tests sans être sur le terrain, il semblait indispensable que le programme puisse utiliser un fichier vidéo en lieu et place de la source à analyser. Il est donc possible de spécifier au lancement du programme le chemin d'un fichier vidéo à utiliser. Il est aussi possible de demander au programme de lire la vidéo en boucle. Sinon, le programme se ferme une fois la vidéo terminée. Il est aussi possible de créer une attente entre chaque frame de la vidéo afin de ralentir sa lecture. A l'inverse, il est possible de choisir de ne pas lire toutes les frames pour accélérer la vidéo. L'attente peut aller jusqu'à une lecture des frames en pas à pas, très utile pour analyser l'effet de changement dans l'algorithme sur une seule image précise. Par défaut, si aucun fichier n'est indiqué, c'est la webcam ou la caméra principale (choisie par l'OS) qui est utilisée. Si aucune caméra n'est disponible, le programme affiche une erreur et s'arrête.

Ensuite, même si la Raspberry PI est munie d'un bon processeur, nous sommes tout de même limité en puissance. Il faudra donc essayer de réduire au maximum le temps que l'algorithme mettra pour détecter les plantons. Un bon moyen de réduire drastiquement ce temps est d'agir sur la taille de la vidéo, respectivement de l'image. En réduisant sa résolution, on diminue le nombre de pixels à traiter et l'algorithme gagne en rapidité. Il faut toutefois faire attention à conserver une résolution suffisamment importante pour réussir à détecter correctement les plantons. La résolution de l'image capturée va donc être amenée à beaucoup changer. Une option permettant de redimensionner l'image source a donc été ajoutée au logiciel. Une option permettant de redimensionner l'image de sortie a aussi été ajoutée dans un soucis de confort visuel. En modifiant certaines constantes dans le code et en recompilant le tout, il est également possible d'activer ou de désactiver l'affichage des différentes fenêtres représentant les différentes étapes de l'algorithme.

4.5.2 Fourchette de couleurs détectées

Afin de grandement simplifier l'étude de l'algorithme et les tests, il a été rendu possible de modifier en temps réel les valeurs de la couleur à détecter lors de la binarisation. Cela permet de pouvoir constater visuellement l'effet de ces changements et ce, de manière dynamique, même lorsque l'image est fichée (mode de lecture pas à pas).



FIGURE 4.3 – Exemple de réglage - Fenêtre binarisation

4.5.3 Taille du noyau des opérations morphologique

Ce sont les opérations morphologiques qui prennent le plus de temps au logiciel pour analyser une image et ce temps dépend grandement de la taille du noyau (ou élément structurant) utilisé. Il est donc nécessaire d'utiliser les noyaux les plus petits possible et pour cela, il est nécessaire de pouvoir modifier ces noyaux et de pouvoir constater visuellement leur effet, tout comme pour les couleurs détectées. Cette possibilité a été implémentée et les tests et les optimisations sont maintenant beaucoup plus simples. De plus, étant donné que la résolution de la source peut changer, une valeur absolue par défaut n'avait pas beaucoup de sens. Les tailles par défaut des noyaux sont donc directement liées à la résolution de l'image. Seul le rapport (valeur relative) par défaut est fixé dans le code.

4.5.4 Distances et tailles des objets

Là encore, il est possible de paramétrier en temps réel la taille minimal des objets détectés ainsi que la distance nécessaire entre chacun d'eux. Pour gérer les différentes perspective, chacune de ces deux options ont deux valeurs : une pour le haut de l'image (objets lointains) et une pour le bas de l'image (objets proches). Entre deux, c'est un rapport proportionnel qui s'applique pour les distances et un rapport exponentiel pour la taille car il s'agit en fait de l'air de l'objet (en nombre de pixel).

4.5.5 Acquisition de données

Comme mentionné plus haut, le logiciel est capable de générer un fichier avec des valeurs de couleurs HSV. Lorsque cette option est activée, à chaque clique de souris dans la

fenêtre affichant la vidéo une ligne est ajoutée au fichier existant (si le fichier n'existe pas il est créé). Une ligne du fichier contient les 3 valeurs (HSV) de la couleur du pixel cliqué.

Il est aussi possible de sauvegarder le temps d'exécution des différentes phases de l'algorithme avec des informations utiles (résolution, taille kernel, etc...). Dans cette idée, un fichier contenant le détail du timing de chaque frame est créé (ou remplacer s'il existe déjà) et permet ensuite de voir si une frame d'une vidéo est particulièrement lente et pourquoi. Une ligne de résumé est aussi ajoutée dans un second fichier (qui est créé s'il n'existe pas) et contient un résumé de l'exécution de l'algorithme sur toute la session (temps moyen de chaque phase de l'algorithme, fréquence maximum des images, etc...) ainsi que l'état des paramètres de détection au moment du lancement du logiciel.

De plus, un autre petit programme, indépendant du programme principal, a été écrit pour permettre de capturer des vidéos depuis la raspberry PI. Ce programme a été utile pour capturer une vidéo lors des tests en laboratoire. Il sera aussi utile pour capturer les données lors des tests sur le terrain.

4.5.6 Logs et debug

Plusieurs options d'affichage de log ont aussi été réalisées. Étant donné que ces options ne sont utiles qu'au moment du développement, il n'a pas été jugé utile d'alourdir le logiciel en les intégrant en temps que réelles options. Ces options sont donc accessibles en modifiant un fichier de constantes et en recompilant le logiciel. Les affichages en console des valeurs/-variables ont été faits avec des macros différentes pour différencier les domaines de debug afin de rendre les informations désirées lisibles en évitant d'avoir beaucoup trop de données inutiles s'affichant en même temps dans la console.

4.6 Tests & Résultats

Cette section décrit deux tests différents effectués pour vérifier le fonctionnement de l'algorithme décrit ci-dessus. Deux points sont importantes à vérifier : Les plantons et uniquement les plantons doivent être reconnus sur un maximum de frame et la Raspberry PI doit pouvoir le faire en un minimum de temps ! Ainsi, les pages suivantes présentent les deux séries d'images correspondantes à chaque étape de l'exécution de l'algorithme. A noté que, pour l'image du résultat de la séparation des objets, c'est la matrice des IDs de chaque pixel qui a été affichée. L'ID 0 donne donc une couleur noir et l'ID 253, une couleur blanche. Ceci explique cette impression de "couleurs inversées". Afin de rendre les différences entre les objets bien visible et puisque seulement quelques objets seront détectés, les IDs des objets ont été incrémenté de 20 en 20 au lieu de 1 en 1. Grâce à cela, les différents plantons apparaissent en différentes nuances de gris.

Tous les tests ont été effectués sur la Raspberry PI 3B+. Le programme utilisé pour ces tests est le programme principal du projet. Pour chaque test, est indiqué la source des images ainsi que les paramètres de l'algorithme utilisés et jugés optimaux pour la vidéo analysée. Dans ce rapport, un résumé des résultats est inclus et commenté. Les résultats détaillés pour chaque image analysées sont donnés en annexe.

4.6.1 Test 1 - Vidéo du champ

Ce premier test correspond à une vidéo capturée par une tablette dans le champ du mandant en début de projet. Dans ce cas où la vision de la caméra est perpendiculaire au

champ, la perspective n'est pas prise en compte. De plus, les couleurs ressortent bien. Elles ressortent si bien que même avec une résolution très faible (ce qui explique la mauvaise qualité de l'image une fois agrandie), la binarisation réussit déjà à très facilement repérer les plantons. Une petite ouverture morphologique est toutefois nécessaire pour bien nettoyer l'image.

Conditions de test

Fichier vidéo : ombreleger.mp4

Frame analysées : 50 : (532 à 630 - uniquement les frames paires)

Résolution d'image : 256x144 (16 :9)

Valeur de couleur acceptées : H : 24 -> 45, S : 63 -> 255, V 0 -> 255

Opération morphologique : Ouverture simple (érosion puis dilatation)

Kernel : Cercle de taille 2 pixels

Taille minimum d'un objet : 36 pixels

Distance minimum entre les objets : 28 pixels

Résultats visuels - Exemple frame n°580



FIGURE 4.4 – Reconnaissance des plantons test 1 - Image source



FIGURE 4.5 – Reconnaissance des plantons test 1 - Binarisation



FIGURE 4.6 – Reconnaissance des plantons test 1 - Érosion



FIGURE 4.7 – Reconnaissance des plantons test 1 - Dilatation



FIGURE 4.8 – Reconnaissance des plantons test 1 - Séparation en objets distincts



FIGURE 4.9 – Reconnaissance des plantons test 1 - Ajout des marqueurs sur l'image source

Résultats généraux

Plantons détectés : 318 / 365 = 87,12%

Nombre de faux positifs moyen par image : 0,00

Temps moyen de la binarisation : 1,939 ms

Temps moyen de l'opération morphologique : 176 us

Temps moyen de la séparation des objets : 6,314 ms

Temps moyen du calcul de la position des objets : 73 us

Temps moyen de la suppression des objets trop proches : 39 us

Temps moyen total pour une frame : 8,541 ms

fréquence moyenne : 117,078 Hz

Frame la plus lente : 594

Temps pour cette frame : 17,573 ms

Fréquence minimum (worst case) : 56,905 Hz

Analyse et conclusion du test 1

Les quelques erreurs sont dues à des plantons au bord de l'image. Ces derniers ne sont pas détectés tant qu'une partie suffisamment grande n'est pas visible. Dès que c'est le cas, ils sont détectés correctement. Ces "erreurs" n'en sont donc pas vraiment. De plus, la résolution étant vraiment très faible, la fréquence de traitement est très élevée, bien plus que nécessaire. On peut donc conclure que, pour cette vidéo, l'analyse d'image fonctionne parfaitement.

4.6.2 Test 2 - Vidéo filmée en laboratoire

Le second test est une vidéo capturée par la caméra du prototype et sa Raspberry PI lors de la phase de test (voir chapitre 7). Ici, la couleur verte imprimée sur papier ressort moins bien que lors du test ci-dessus. Cependant, une résolution élevée ne donnait pas de meilleur résultats. La résolution utilisée est donc faible ici aussi. La caméra est inclinée à environ 45°. Il est donc nécessaire de prendre en compte la perspective comme expliqué plus tôt dans ce chapitre. Avec cette angle, l'air d'un objet proche est 4 fois plus grande que celle d'un objet lointain. Les distances, par contre, sont seulement doublées.

Conditions de test

Fichier vidéo : videolabo.avi

Frame analysées : 50 : (1601 à 1650)

Résolution d'image : 320x240 (4 :3)

Valeur de couleur acceptées : H : 34 -> 66, S : 66 -> 255, V 0 -> 255

Opération morphologique : Ouverture simple (érosion puis dilatation)

Kernel de l'érosion : Croix de taille 2 pixels

Kernel de la dilatation : Croix de taille 3 pixels

Taille minimum d'un objet lointain : 11 pixels

Taille minimum d'un objet proche : 46 pixels

Distance minimum entre les objets lointains : 30 pixels

Distance minimum entre les objets proches : 69 pixels

Résultats visuels - Exemple frame n°1630

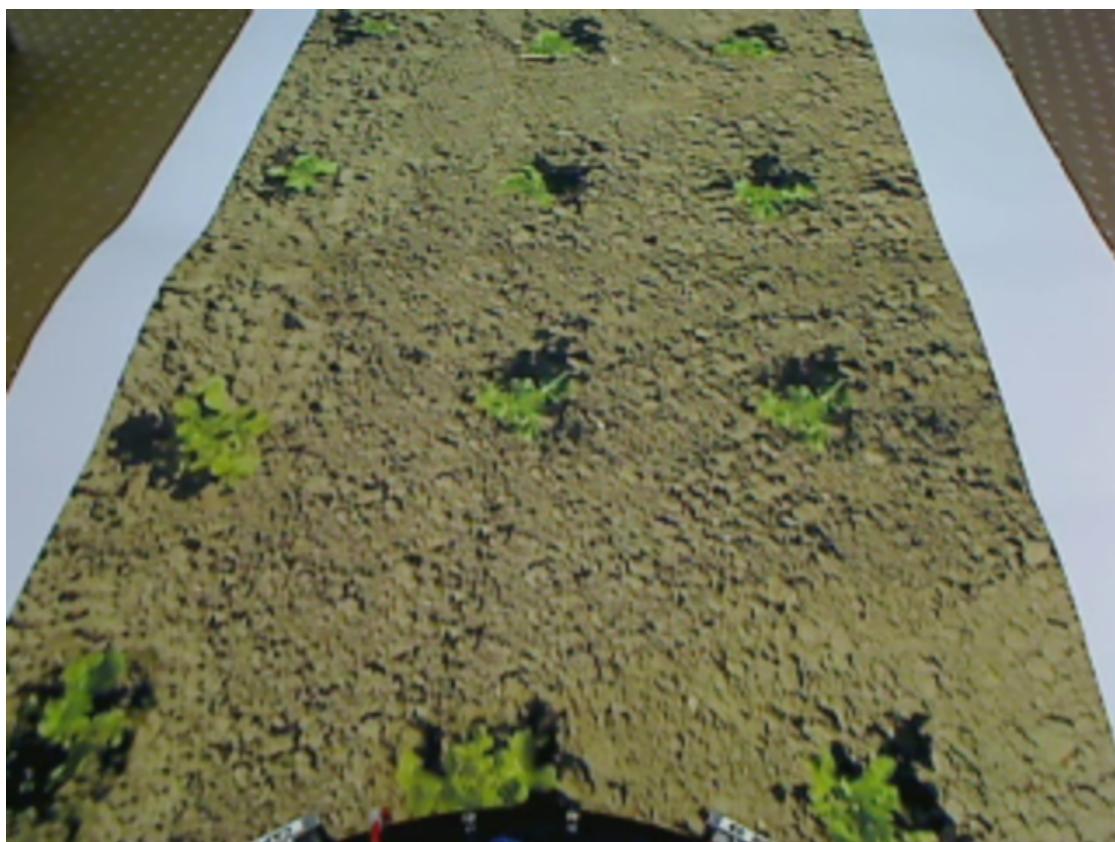


FIGURE 4.10 – Reconnaissance des plantons test 2 - Image source

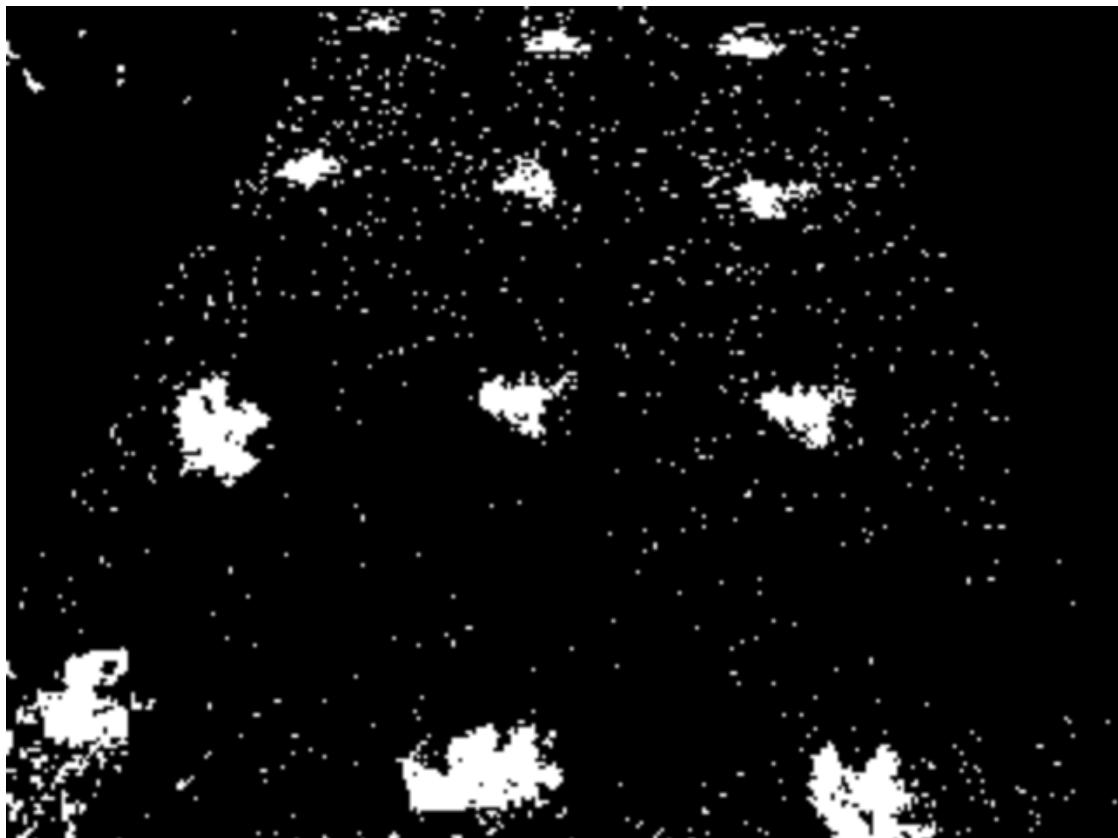


FIGURE 4.11 – Reconnaissance des plantons test 2 - Binarisation

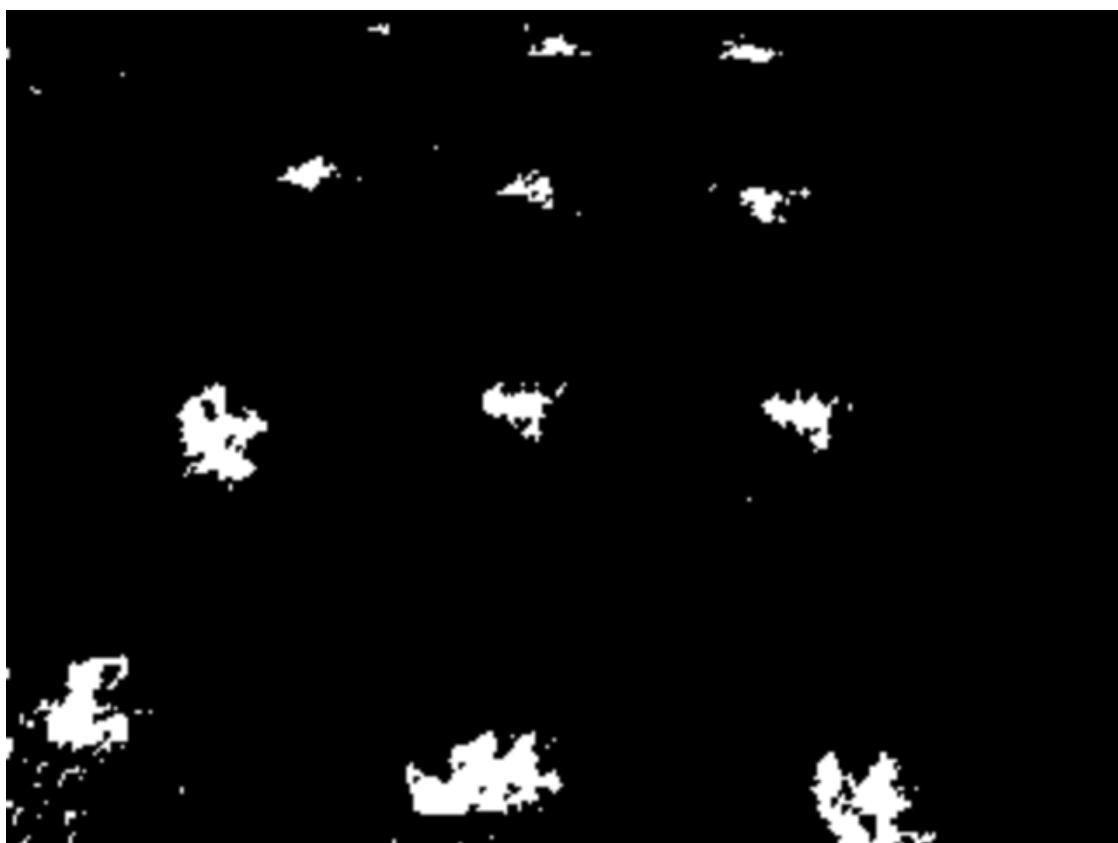


FIGURE 4.12 – Reconnaissance des plantons test 2 - Érosion

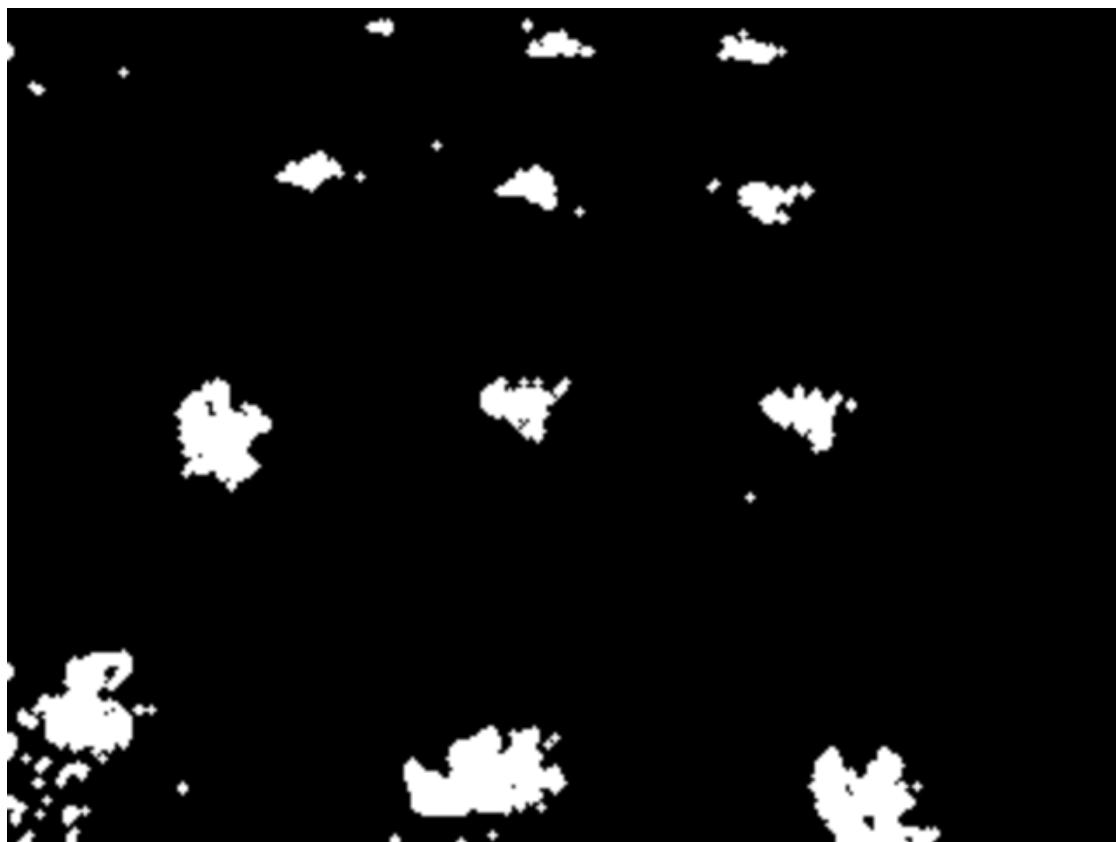


FIGURE 4.13 – Reconnaissance des plantons test 2 - Dilatation



FIGURE 4.14 – Reconnaissance des plantons test 2 - Séparation en objets distincts

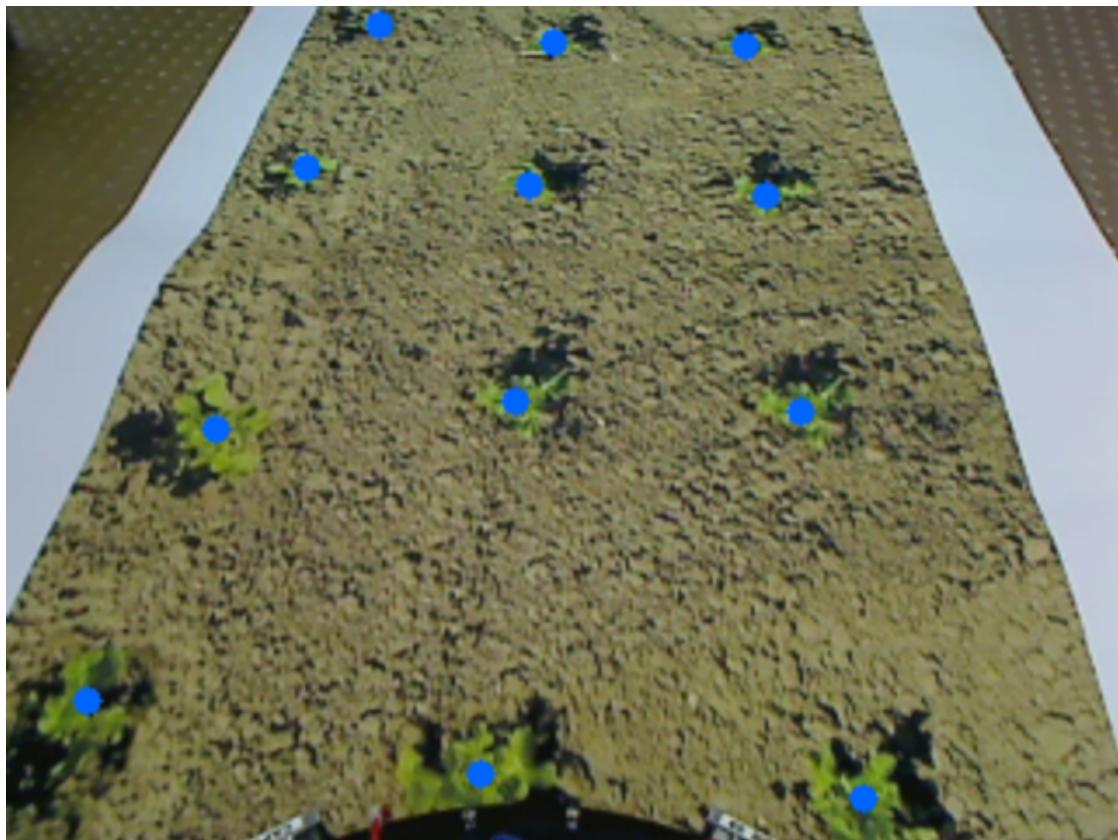


FIGURE 4.15 – Reconnaissance des plantons test 2 - Ajout des marqueurs sur l'image source

Résultats généraux

Plantons détectés : 495 / 498 = 99,40%

Nombre de faux positifs moyen par image : 0,46

Temps moyen de la binarisation : 3,941 ms

Temps moyen de l'opération morphologique : 2,976 ms

Temps moyen de la séparation des objets : 12,723 ms

Temps moyen du calcul de la position des objets : 79 us

Temps moyen de la suppression des objets trop proches : 52 us

Temps moyen total pour une frame : 19,770 ms

fréquence moyenne : 50,581 Hz

Frame la plus lente : 1605

Temps pour cette frame : 38,115 ms

Fréquence minimum (worst case) : 26,236 Hz

Analyse et conclusion du test 2

Là encore, les plantons qui ne sont pas détectés sont ceux dont une trop petite partie est visible sur l'image pour qu'ils entrent dans les critères de détections. En ce qui concerne les faux positifs, ils sont détectés en dehors de l'image imprimée. Pour avoir de meilleurs résultats, il aurait suffit de couvrir l'extérieur de la feuille de blanc ou d'imprimer une image plus grande. Toute comme le test précédent, les plantons sont donc parfaitement détectés. En ce qui concerne le temps d'exécution, il est certe plus important que pour le test précédent mais reste tout à fait acceptable.

4.6.3 Conclusion des tests

L'algorithme fonctionne parfaitement dans les cas testés en laboratoire. Cependant, il est nécessaire d'effectuer des tests en situations réelle pour pouvoir affirmer que tout fonctionne vraiment. Nous pouvons toutefois imaginer que les couleurs réels (intensité et lumière) se rapprocheront plus de celle du test n°1. De plus, une meilleure caméra devrait encore améliorer les résultats et assurer le succès du projet.

Par contre, dans le cas où une meilleure résolution venait à être nécessaire, il faudrait certainement optimiser l'algorithme (voir-dessous).

4.6.4 Améliorations possibles du logiciel

Il serait bien de modifier un peu la gestion du dépassement de la limite de récursion dans la fonction analyzePixel. Actuellement, lorsque la limite est atteinte, certains pixels sont ignorés mais l'algorithme continue sans forcément remonter dans la récursion et réatteint la limite juste après. Il serait préférable, lorsque la limite est atteinte, de remonter toute la récursion (les pixels visités auront déjà eu le bon ID assigné) puis de recommencer à ce moment là une nouvelle récursion sur les pixels encore non analysés du même objet.

Actuellement, le programme pour capturer une vidéo est un programme différent du programme principal. Il serait souhaitable que la capture de vidéo devienne une options supplémentaire du programme principal, afin de pouvoir garder une trace vidéo des résultats des tests effectués.

4.7 Futurs tests et optimisations

Une fois le prototype du vrai robot construit et les cartes installées avec l'écran et la webcam, il faudra faire de vrais tests sur le terrain. La position et l'inclinaison de la caméra sur le robot jouera un grand rôle. Heureusement, le programme actuelle peut facilement être adaptés : il est très facilement paramétrable et permet d'enregistrer certains résultats de tests automatiquement.

Si besoin, pour améliorer la qualité de la détection de l'algorithme, la première chose à essayer serait de modifier les paramètres de détection et plus spécialement la taille des kernels. Il peut aussi être utile de faire des tests en utilisant un format de couleur différent, par exemple le format HUY (YCbCr). Si la qualité de la détection n'est pas améliorée grâce à ça et/ou qu'elle n'est toujours pas suffisante, se tourner vers les réseaux de neurones serait une option mais signifierait une grosse charge de travail supplémentaire.

Concernant la vitesse d'exécution de l'algorithme, voilà 4 idées pour l'améliorer :

Utiliser plusieurs threads : La Raspberry pi possède 4 coeurs. Actuellement, cette algorithme n'en utilise qu'un seul. A la base, il avait été prévu d'utiliser plusieurs caméras et donc, d'utiliser un seul thread pour un flux vidéo. Dans le cas où le projet final ne conserve qu'une seule caméra, optimiser l'algorithme en utilisant plusieurs threads peut être une bonne idée. Il faudra toutefois faire attention aux problèmes de concurrence. De plus, il n'est pas très utile de traiter plusieurs images en parallèle car cela ne réduira pas le temps de réaction du robot (latence). Ce qu'il faudrait réussir à faire est donc de diviser les gros temps de traitement pour une seule image en plusieurs threads, notamment les opérations morphologiques et l'algorithme de séparation des objets. Malheureusement, les opérations morphologiques sont des fonctions natives d'openCV. Difficile donc de les utiliser différemment. Ce serait alors uniquement la

moitié, voir le tiers du temps de traitement qui pourrait être amélioré et cela demanderait un travail conséquent. Par certain donc que ce travail en vaille la peine.

Utiliser le GPU au maximum : Nativement, OpenCV est sensé utiliser le GPU disponible.

La Raspberry PI 3B+ en possède un mais rien n'a été fait pour être certain que ce dernier est utilisé. Il l'est probablement pour les opérations morphologiques qui sont des fonction d'OpenCV mais il est presque certain que l'algorithme de séparation des plantons ne l'utilise pas. L'utilisation du GPU pourrait donc être une bonne piste à étudier pour améliorer les performances du système.

L'overclocking : Augmenter les performances de la Raspberry PI au delà des marges de sécurité fixée par le fabricant peut être une idée mais là encore, cela demande beaucoup de recherche et de prise de risque pour une résultat qui risque d'être mitigé. De plus, il faudrait probablement modifier le design mécanique du boîtier pour y inclure un système de refroidissement actif.

Utiliser une carte plus performante : Cette option semble clairement la meilleure. Comme décrit dans le chapitre précédent, la Banana PI ou la Tinker Board sont des alternatives bien plus puissantes et même si leur communauté est plus réduite que celle de la Raspberry PI, porter le projet sur une des ces boards ne devrait pas être très difficile. En effet, elles possèdent toutes les deux un OS dédié et fonctionnel et le support du fabricant est disponible en cas de besoin.

Chapitre 5

Suivis d'une ligne dans un champ

5.1 Gestion des roues

5.1.1 Moteurs

Il était initialement prévus que les moteurs du robot final se commandent par des PWMs. La carte d'extension PWM avait été acquise dans cette optique. Finalement, les moteurs utilisés seront des moteurs pas à pas. Il existe des drivers pour ces moteurs. Il semblerait que ces drivers se commandent avec uniquement des signaux logiques. Si c'est effectivement le cas, la carte PWM peut être utilisée comme simple sortie logique et devrait suffire à complètement le nombre de sortie GPIO de la raspberry PI.

Pour commander le variateur qui alimente les moteurs de modélisme du prototype (voir chapitre suivant), il faut envoyer périodiquement une pulse entre 1ms et 2ms. Une pulse de 1,5ms indique au variateur de stopper le moteur, une pulse de 1ms, lui demande de la faire tourner en arrière à sa vitesse maximum et une pulse de 2ms, de le faire tourner en avant à sa vitesse maximum. Entre ces valeurs, la puissance donnée au moteur est plus ou moins linéaire.

5.1.2 Contrôle de la carte PWM

La carte PWM se raccord à la Raspberry PI par le bus I2C (le pinning des GPIOs de la Raspberry PI est donné en annexe). Pour dialoguer avec la carte PWM, il a fallut activer le bus I2C de la Raspberry PI grâce à la commande `raspi-config`. Ensuite, il a suffit d'utiliser le driver mis à disposition par Raspbian. Pour les 1er tests, un programme permettant de simplement lire et écrire sur le bus I2C a été écrit. Ce petit programme a été utile pour paramétrier la carte PWM et vérifier son bon fonctionnement. Sans entrer dans les détails de toutes les possibilité du circuit PCA9685 (circuit au cœur de la carte PWM, dont le datasheet est en annexe), l'idée est de paramétrier un registre de comparaison. Un compteur à l'interne du circuit tourne en boucle. Lorsque ce dernier atteint la valeur de comparaison paramétrée, la sortie du PWM tombe à 0 et le compteur continue. Lorsque le compteur arrive à sa valeur limite, il recommence à compter à partir de 0 et la sortie du PWM remonte à 1. Ainsi, la fréquence du PWM est fixe mais il est possible de changer le temps à l'état haut (et donc le rapport cyclique). Cette fréquence est de 200Hz. La seule façon de changer cette fréquence est d'utiliser un clock externe mais, malheureusement, la carte PWM n'a pas prévu d'accès à la pin "clock-source" du circuit PCA9685. Mais ce n'est pas bien grave puisque la fréquence convient pour une utilisation avec des moteurs de modélisme et que ce n'est pas des PWMs qui seront utilisés dans la version finale du robot. Pour plus de détail sur les paramètres configurés du circuit PCA9685, se référer directement au code ou au datasheet en annexe.

5.1.3 Déplacements manuels

Avant de pouvoir rendre le robot autonome, il est nécessaire de pouvoir le faire se déplacer manuellement. Pour cela, une jostick low-cost fait très bien l'affaire. Le noyau linux contient un driver inclus prêt à l'emploi[40]. De plus, internet contient beaucoup d'exemple d'utilisation. Pour gagner du temps, nous sommes partis d'un exemple[41] de Jason White qui affiche en console la touche du joystick appuyée. Ce code a ensuite été modifié pour réussir à contrôler les moteurs avec le joystick. Avec deux roues motrices, il y a deux possibilités de voir le déplacement :

- Mode de déplacements semblables à celui d'une voiture, c'est à dire que le robot ne bouge pas si l'on met pas de "gaz" (Mode 0)
- Mode de déplacement où il est possible de tourner sur soi-même, mais dans ce cas, la marche arrière est inversée (Mode 1)

Les deux modes sont intéressants et pour ne pas devoir choisir entre les deux, les deux ont été codés. Une touche du joystick permet de passer d'un mode à l'autre (voir le manuel de l'utilisateur en annexe). Pour mieux comprendre comment réagissent les roues en fonction de la commande dans chacun des deux modes, voici deux petits tableaux qui résument leur fonctionnement. Chaque case correspond à un exemple de commande. Dans chaque case est indiqué l'effet qu'à la commande sur chacune des deux roues (flèche de gauche pour la roue gauche et flèche de droite pour la roue droite). Un trait signifie que la roue en question est arrêtée. D'une case à l'autre, le passage est linéaire.

Mode0 (mode "voiture")

	Gauche	Tout droit	Droite
Avancer	↑	↑↑	↑—
Sur place	—	—	—
Reculer	↓	↓↓	↓—

FIGURE 5.1 – Gestion des roues - Mode n°0

Mode1 (mode "tourner sur place")

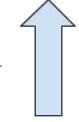
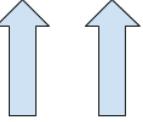
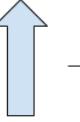
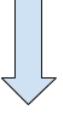
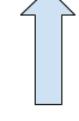
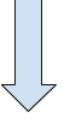
	Gauche	Tout droit	Droite
Avancer			
Sur place			 
Reculer			

FIGURE 5.2 – Gestion des roues - Mode n°1

5.2 Pilote automatique

Le but final est que le robot se déplace tout seul, en suivant la ligne du champ, à une vitesse très faible. Par "suivre la ligne" on entend deux choses : Être bien aligner avec les plantons (angle) et être bien centré sur la ligne du milieu (offset proche de 0 sur l'axe X). Le pilote automatique peut être activer ou désactiver en appuyant une touche du joystick (voir le manuel de l'utilisateur en annexe).

5.2.1 Reconnaissance de la direction à prendre

Pour les 1ers tests en laboratoire, il a été définit que la robot devait être positionner en face de la ligne au début et avancer en corrigeant sa trajectoire. Il a également été choisis que la ligne serait suivie grâce aux plantons détectés lors de la reconnaissance (voir chapitre 4). Pour rester droit, il faut rechercher les plantons appartenant à la 1ère ligne horizontale devant le robot et s'assurer d'y être bien perpendiculaire. Pour se faire, on trie d'abord les plantons par leur coordonnée Y. Ce tri est en réalité presque déjà fait lors de la reconnaissance puisque que le tableau d'objet est rempli en commençant en haut de l'image. Ensuite, on récupère uniquement les 3 derniers objets (les plus proches) et on vérifie si leur coordonnées Y sont proches. Si ne c'est pas le cas (coordonnées Y trop différentes), on considère que les plantons ne font pas partie de la même ligne et cette frame est ignorée : on attendra la frame suivante pour de réaligner. On aurait aussi pu récupérer un 4ème planton pour augmenter les chances de trouve une ligne de planton. C'est une idée à garder en tête si le code actuel ne donne pas satisfaction. Si les 3 plantons ont des coordonnées Y proches, on considère alors qu'ils forment la 1ère ligne de plantons devant le robot. La figure 5.3 montre un exemple d'une ligne détectée et validée (les plantons jaunes sont ceux que l'algorithme

analyse). La figure 5.4 montre un exemple d'une ligne qui a été refusée par l'algorithme. Cette frame sera ignorée.

Il est maintenant possible s'aligner sur cette ligne en comparant la coordonnée Y du planton de gauche (plus petite coordonnée X des trois) avec celui de droite (plus grande coordonnée X des trois). Si leur différence est plus grande qu'une constante paramétrable, le robot doit se réaligner. Pour se faire, il tourne légèrement en bloquant une de ces roues pour un temps proportionnel (rapport constant paramétrable) à l'angle à corriger. Il répète cette opération tant qu'il ne considère pas être aligné. Une fois qu'il est aligné, il vérifie alors qu'il est bien centré. Pour cela, il effectue la moyenne des 3 cordonnées X des plantons de la 1ère lignes. Si cette moyenne diffère trop (limite constante paramétrable) de la moitié de la largeur de l'image, le robot lance sa séquence de recentrage :

1. Effectuer un virage à 90°dans la direction du centre = tourner sur place pendant un temps constant paramétrable, à une vitesse constante
2. Avancer lentement de la distance à corriger = avancer pendant un temps proportionnel (rapport constant paramétrable) à la distance à corriger
3. Effectuer un virage à 90°inverse au 1er
4. Vérifier l'alignement et le corriger si nécessaire
5. Vérifier le positionnement horizontale et relancer une séquence de recentrage si nécessaire

Lorsque le robot effectue sa séquence de recentrage, tout autre forme de modification de trajectoire est désactivée.

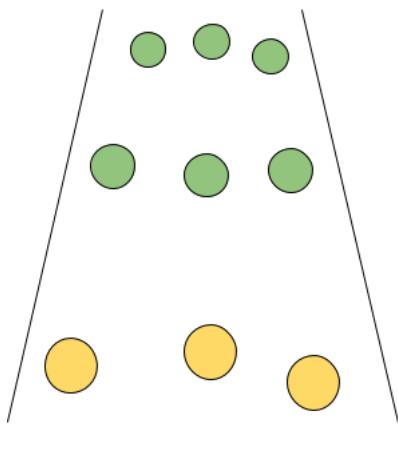


FIGURE 5.3 – Exemple de détection de ligne validée

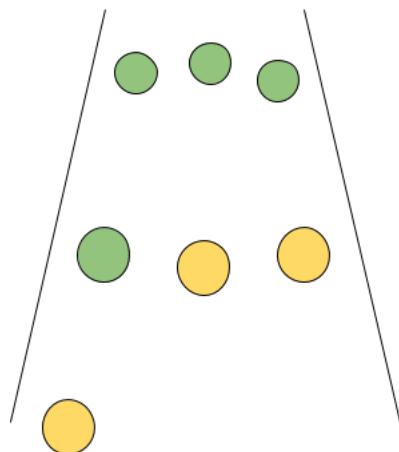


FIGURE 5.4 – Exemple de détection de ligne refusée

5.2.2 Autre moyens

Les plantons n'étant pas aligné horizontalement chez le maraîcher, il faudrait probablement se baser sur les lignes verticales plutôt que les lignes horizontales. Si la caméra est perpendiculaire au sol, rien de plus facile. Il suffit de détecter les lignes verticales puis de vérifier la différence des coordonnées X du planton le plus proche et du plus lointain. Si cette différence dépasse une certaine constante paramétrable, il faut corriger la trajectoire.

Dans le cas où, la caméra n'est pas perpendiculaire, il faut utiliser la perspective. Une fois les lignes verticales de plantons détectées, on peut les prolonger virtuellement et vérifier

leur point d'intersection. Si la coordonnée X de ce point s'écarte du centre de l'image de plus d'un constante paramétrable, il faut corriger la trajectoire.

En réalité, il devrait aussi être possible de détecter les bords du champ et d'utiliser la perspective de la même manière que pour les lignes de plantons.

5.2.3 Fin de la ligne

Lorsque que le robot détecte moins de 3 plantons, on considère qu'il est arrivé à la fin de la ligne. Dans ce cas, pour cette première version, le robot s'arrête simplement et le pilote automatique se désactive. Il peut ensuite être réactivé en appuyant sur la touche du joystick correspondant. Il serait aussi possible de le faire faire demi-tour et recommencer dans l'autre sens. Cette version sera codé si le temps le permet

5.3 Tests, état actuel et travail restant

Pour le moment, le code de l'algorithme est toujours en cours d'écriture. Il n'a donc pas pu être complètement testé. Il ne reste normalement que quelques heures de travail et il devrait donc être très prochainement fonctionnel. Par contre, l'avance lente et automatique fonctionne et peut être activée ou désactivée avec le joystick. Les moteurs de modélisme (ou les variateurs) montés sur le prototype ne sont cependant pas très précis et il est difficile de les faire tourner lentement de manière continue et à la même vitesse. Les valeurs des PWM envoyées aux deux roues ont été réglées indépendamment pour donner le meilleur résultats possible. Malheureusement, la vitesse des roues et la conservation du cap est très dépendant de la charge de la batterie. Attention donc à avoir une batterie complètement chargée lors des tests. Heureusement, les moteurs pas à pas prévus sur le robot final sont beaucoup plus précis.

Il est aussi nécessaire de prévoir un moyen de recevoir des informations de la commande du bras. En effet, si ce dernier n'as pas suffisamment de temps d'arroser il doit pouvoir indiquer au thread qui gère les roues de s'arrêter.

Chapitre 6

Généralités logiciels

6.1 Fonctionnement général

6.1.1 Structure logiciel

Le schémas ci-dessous décrit comment les différentes parties du programme interagissent entre-elles. Les périphériques d'entrées sont en vert et les périphériques de sorties sont en rouge. Les éléments grisés correspondent à des fonctionnalités qui n'ont pas encore été implémentées mais qui sont prévues. Certaines flèches blanches actuellement en place seront supprimées une fois que les parties grises seront implémentées. Il s'agit d'une interface utilisant l'écran tactile et permettant de faire quelques réglages "hauts niveaux" comme de choisir le type de planton à détecter, démarrer l'enregistrement d'une vidéo ou encore démarrer/arrêter le pilote automatique.

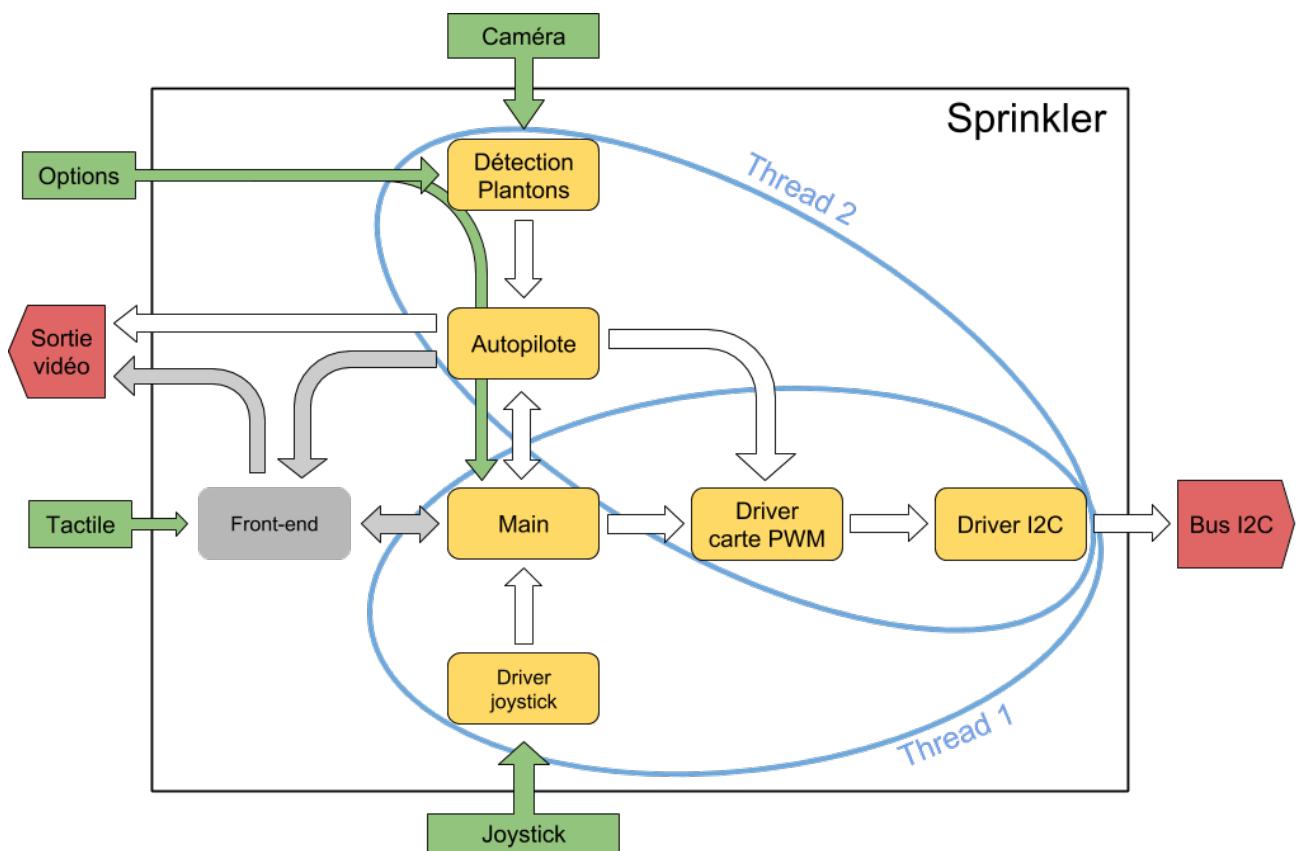


FIGURE 6.1 – Schéma complet de la structure du programme

6.1.2 Rôle des threads et concurrence

Ce programme fonctionne actuellement sur deux threads. Le premier thread (le thread principal), commence par parser les arguments en entrées (options) puis initialise le joystick, le bus I2C et la carte PWM. Il démarre ensuite le pilote automatique sur un deuxième thread et attend de recevoir un événement provenant du joystick. Lorsqu'il en reçoit un, il demande l'accès au bus I2C via un mutex et le conserve jusqu'à ce que les commandes du joystick soient relâchées. Pour rappel, un mutex est un objet accessible depuis plusieurs threads qui permet d'éviter les problèmes de concurrence. Dans notre cas, il sert à deux chose : il empêche qu'une écriture sur le bus I2C soit interrompue par une autre écriture et il empêche le pilote automatique de reprendre la main lorsque le joystick pilote robot en mode manuel. Il est toutefois préférable de désactiver le pilote automatique (grâce à la touche dédie du joystick) avant de passer en mode manuel.

Le second thread est celui du pilote automatique. Ce dernier commence par initialiser la capture vidéo. Il initialise ensuite les options de détection qui étaient dépendantes de la taille de l'image, crée les fenêtres pour visionner la détection des plantons puis démarre l'algorithme de reconnaissance des plantons. Une fois les plantons détectés, il calcule la correction à effectuer sur la trajectoire. Si une correction s'avère nécessaire, il demande l'accès au bus I2C et change les valeurs des PWM générés en écrivant dans les registres de la carte PWM via I2C. Dans la prochaine version du programme, ce thread devra aussi gérer la position du bras d'arrosage en s'alignant sur le planton de la ligne centrale.

6.2 Pré-requis

Voici la liste des pré-requis pour faire fonctionner les logiciels de ce projet :

- Raspberry PI modèle 3B+ avec un système d'exploitation Linux compatible Debian (idéalement Raspbian)
- OpenCV v.3.2.0 installé
- Source de flux vidéo disponible (caméra, webcam ou fichier vidéo)
- Protocole I2C activé

En option :

- Joystick ou gamepad connecté pour une commande manuel
- Écran connecté via HDMI ou flux vidéo via SSH pour afficher l'image

Toutes les informations nécessaires, les marches à suivre pour l'installation, les numéros exactes de version et les liens pour se procurer les sources logicielles se trouvent dans un fichier nommé "Installation et configuration" en annexe de ce rapport. De plus, un CD contenant le code, les exécutables, les résultats des tests et toute la documentation, ainsi que des cartes SD toutes installées pour raspberry PI sont également jointent au projet.

6.3 Compilations

Toutes les sources du programme se trouve dans le dossier dev du projet. Lors de la compilation, il faut demander au compilateur d'utiliser les librairie opencv. Pour ce faire, il faut ajouter le complément suivant à la commande de compilation : '`pkg-config --cflags --libs opencv`'. Il faut également ajouter l'option `-lpthread` pour indiquer au compilateur que les

pthreads sont utilisés dans le code. De plus, bien que le code soit écrit en C, il est nécessaire d'utiliser un compilateur C++ car une option a dû être écrite en C++.

Les Makefile inclus dans chaque dossier effectuent tout ça automatiquement. Il suffit de les appeler avec la commande *make*. Il y a un fichier main et un Makefile par dossier. Cela permet de compiler une seule partie du code. Contrairement à ce que l'on pourrait imaginer, les Makefile se sont inclus les uns dans les autres. Ils sont tous indépendants les uns des autres et compilent les fichiers nécessaires pour faire fonctionner le programme main de leur dossier respectif. A l'exception du fichier main général (dans le dossier dev), les autres fichiers main contiennent une version antérieur d'un programme pour tester les fonctions des autres fichiers du dossier. Ils sont conservés ici à titre d'archive uniquement et leur compilation n'est plus garantie car les fichiers qui y étaient inclus ont été modifiés.

6.4 Options et paramètres

Comme décrit dans les chapitres précédents, un total de 25 paramètres différents sont disponibles au lancement du programme principal. Les paramètres se gèrent en passant des paramètres en console. La liste est décrite dans le manuel d'utilisateur mais peut aussi être affiché avec le paramètre *-r*. Cette commande vous indiquera aussi la fonction des touches du joystick.

Pour les options de debug et de développement, il faut aller modifier le fichier "options.hpp" disponible dans le dossier dev et recompiler le programme. Attention toutefois à ne pas activer trop d'options de debug à fois car la console risque de devenir illisible. C'est aussi grâce à ce fichier qu'il est possible de créer des fichiers csv contenant les timings de l'algorithme de détection ou les valeur HSV des couleurs cliqué dans la fenêtre du flux vidéo.

6.5 Démarrer un logiciel automatique après le boot

Actuellement, le programme démarre automatique avec les options par défaut à la fin du boot de la raspberry PI. Il existe deux méthodes pour démarrer une application au démarrage de la carte :

- Démarrer en même temps que le système (ne fonctionne pas pour les applications GUI (applications graphiques))
- Démarrer avec les applications utilisateur, une fois que le bureau a été chargé en mémoire (seule option valide pour les applications graphiques tel que notre programme)

6.5.1 Démarrage avec le système

Lors de son démarrage le système exécute le script suivant : */etc/rc.local*. Il suffit donc de modifier ce script en ajoutant un programme à démarrer. Il faut toutefois faire bien attention à être certain que le script ne crée pas une erreur ou une boucle infinie. N'oubliez pas d'ajouter un "&" à la fin de la ligne pour lancer le programme en arrière plan et n'hésitez pas à essayer de lancer le script manuellement pour vérifier son bon fonctionnement. Pour simplifier l'accessibilité, un autre script faisant partie du projet est appelé par celui-ci. Le script appelé est */home/pi/sprinkler/startScripts/startOnRaspbian*. Actuellement, étant donné que la dernière version du programme forme un tout, le script est vide. Il peut toutefois toujours être utile pour démarrer par exemple un programme critique qui assurerait la sécurité. A

noter que les programmes démarrés de cette manière tournent sur le système directement et ne sont pas dépendant d'un quelconque utilisateur. Il est donc nécessaire d'avoir des droits d'admin (sudo) pour les arrêter.

6.5.2 Démarrage en mode utilisateur

A la fin du démarrage du bureau GUI, le système exécute le script suivant : `/home/pi/.config/lxsession/LXDE-pi/autostart`. Comme pour la méthode ci-dessus, il suffit de modifier ce script pour ajouter un programme à démarrer. Les recommandations sont les même que ci-dessus. Là encore, un appel vers un autre script a été écrit. Cet autre script est le suivant : `/home/pi/sprinkler/startScripts/startOnDesktop`. Actuellement, ce dernier script lance le programme général avec une résolution de 320x240, ce qui est petit mais suffisant. Ce programme ainsi démarré fait automatiquement avancer le robot en mode autopilot. Pour désactiver l'autopilot, il faut appuyer sur la touche correspondante du joystick (actuellement la touche 4). Pour arrêter la détection des plantons (qui continue de fonctionner même avec le pilot automatique désactivé), il faut appuyer sur la touche "q" quand une des fenêtre du logiciel est activé. Si aucune fenêtre n'est activée ou si l'on désire arrêter complètement le programme (et non pas juste la détection et le pilote automatique), il est alors nécessaire de kill le process. Pour ce faire, on utilise la commande `top` pour afficher la liste des process. Une fois l'ID du process connu, on utilise la commande `kill` suivi du l'ID en question.

6.6 Améliorations possibles

Le programme actuel est codé en C. Utiliser le C était un moyen d'être certain d'être compatible avec n'importe quelle board et de limiter au maximum les pertes de performance. Malheureusement, plus le travail avançait, plus il devenait important de passer au C++. Une des options importantes codées n'était d'ailleurs disponible qu'en C++. (Il s'agit des curseurs permettant de modifier les valeurs de détections de manière dynamique). De plus, nous avons appris récemment que les fonctions C d'OpenCV sont dépréciées depuis plusieurs années et que leur support n'est plus assuré. Au vu de la taille du projet qui commence à prendre de l'ampleur, il serait aussi beaucoup plus propre de pouvoir faire de l'orienté objet. Migrer du C vers le C++ serait donc très bienvenu et rendrait le code plus lisible tout en assurant sa pérennité.

Actuellement, comme expliqué ci-dessus, si la fenêtre n'est pas le seul moyen de kill le processus général du programme lancé au démarrage est de le kill de manière brute en ligne de commande. Il faudrait trouver une alternative pour le fermer.

Même si beaucoup d'options sont déjà accessibles de manière dynamique, ce n'est pas le cas de toutes les options. Les options de debug par exemple, nécessitent une recompilation. Il pourrait être pratique de créer un fichier de config qui serait lu au démarrage du fichier et qui contiendrait toutes les constantes écrites dans le fichier "options.hpp".

Chapitre 7

Prototype et banc de test

7.1 Une mauvaise nouvelle

A environ 2/3 du temps de travail alloué pour ce projet, nous avons appris que la partie mécanique du projet ne serait pas en mesure de fournir un prototype, même sommaire ou à échelle réduite, du robot final. Il a donc fallu réorganiser le planning pour y inclure la conception d'un prototype permettant de démontrer le fonctionnement des l'algorithme décrit dans ce rapport. Même si cette tâche ne correspond pas vraiment à un travail informatique, elle était toutefois nécessaire. Heureusement, plusieurs autres personnes et département se sont rendu utiles et ont permis de créer quelque chose de correct dans les temps. Le projet a par contre indubitablement pris du retard par rapport à ce qui était prévu à l'origine et une partie des fonctionnalités qui étaient décrites comme "minimales" au chapitre 2 n'auront finalement pas été complètement réalisées au terme du temps alloué à ce travail.

7.2 Le prototype

7.2.1 le reds bot

Pour la base de ce prototype, nous avons utilisé un ancien robot développé dans l'institut : le reds bot. Le reds bot était un robot qui était utilisé lors des portes ouvertes de l'école. Quelques éléments superflus ont été retirés et des collaborateurs de l'atelier de mécanique ont adapté le robot pour qu'il puisse accueillir l'écran avec la raspberry PI et la carte PWM visé derrière. Ils ont aussi permis de surélever la webcam. Un prototype de bras a également été créé un autre atelier et a été monté sous le robot. Finalement, un électronicien de l'école s'est occupé de toute la partie câblage et a ajouté une petite led à l'avant du robot permettant de simuler l'activation de l'arrosage.

Légende des figures 7.1 et 7.2 :

- | | |
|---------------------|-------------------------|
| 1. Écran | 6. Carte d'alimentation |
| 2. Carte PWM | 7. Webcam |
| 3. Raspberry PI | 8. Prototype de bras |
| 4. Led | 9. Moteur bras axe X |
| 5. Variateur moteur | 10. Moteur bras axe Y |

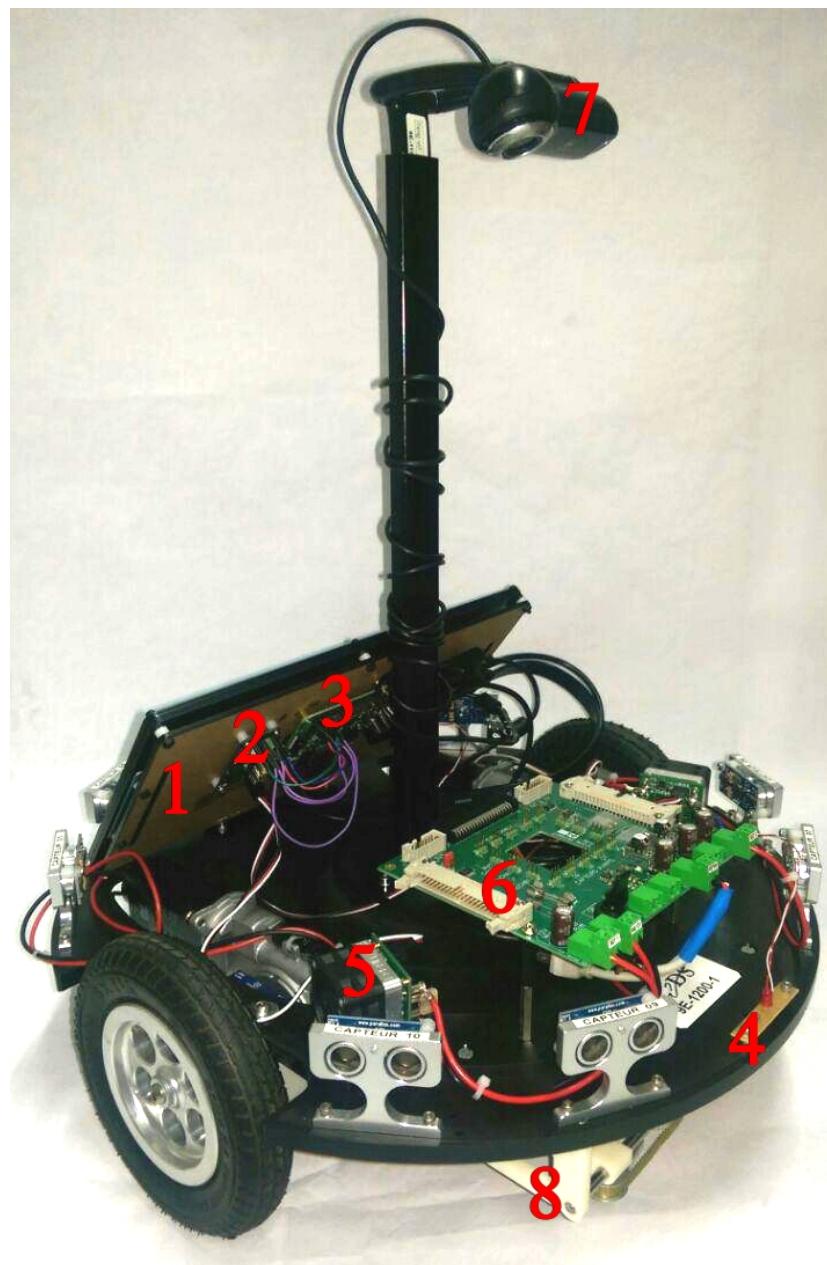


FIGURE 7.1 – Prototype

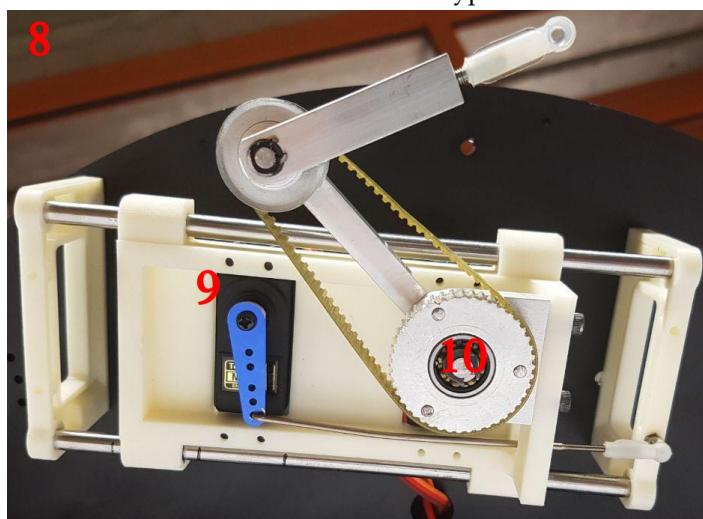


FIGURE 7.2 – Prototype de bras d’arrosage

7.2.2 Prototype du bras

La figure 7.2 montre le prototype d'une bras utilisant 2 moteurs. Une pour l'axe X, un pour l'axe Y. Initialement, il était prévu que les bras fonctionnent avec des cordonnées cylindriques. C'est à dire, des longueurs fixes (comme dans le prototype actuel) mais avec des angles indépendants. Finalement, la conception mécanique a là aussi changé d'avis et a opté pour quelque chose de plus facile à commander. Dans cette nouvelle version du bras, un seul moteur commande les deux angles grâce à une courroie et un rapport de roue rendue de 1/2. Avec cette méthode, une petite roue est toujours 2x plus grande que celle de la grande et le bout du bras reste donc toujours aligné avec la fixation à l'origine. C'est cette même méthode qui est monté sur le prototype. Petit inconvénient tout de même, le bras ayant été récupéré d'un autre robot, il est un peu petit par rapport à la taille du redsbot. Il suffit donc de commander les deux servomoteurs indépendamment. Petite mise en garde néanmoins : la position du bout du bras n'est pas proportionnel à l'angle de la première roue dentée. Son déplacement est sinusoïdal et non linéaire.

7.2.3 Alimentation

La carte d'alimentation (Chiffre rouge N°6 sur la figure 7.1) est celle du redsbot. Elle crée toutes les tensions dont nous avons besoin. Elle alimente les moteurs des roues et des bras ainsi que la raspberry PI et son écran. Le courant délivré n'est pas vraiment suffisant et il arrive que, lors d'un pique de courant la raspberry PI subisse une chute de tension et redémarre. Malgré tout, cela sera suffisant pour une 1ère série de test en attendant le robot final.

7.3 Le banc de test

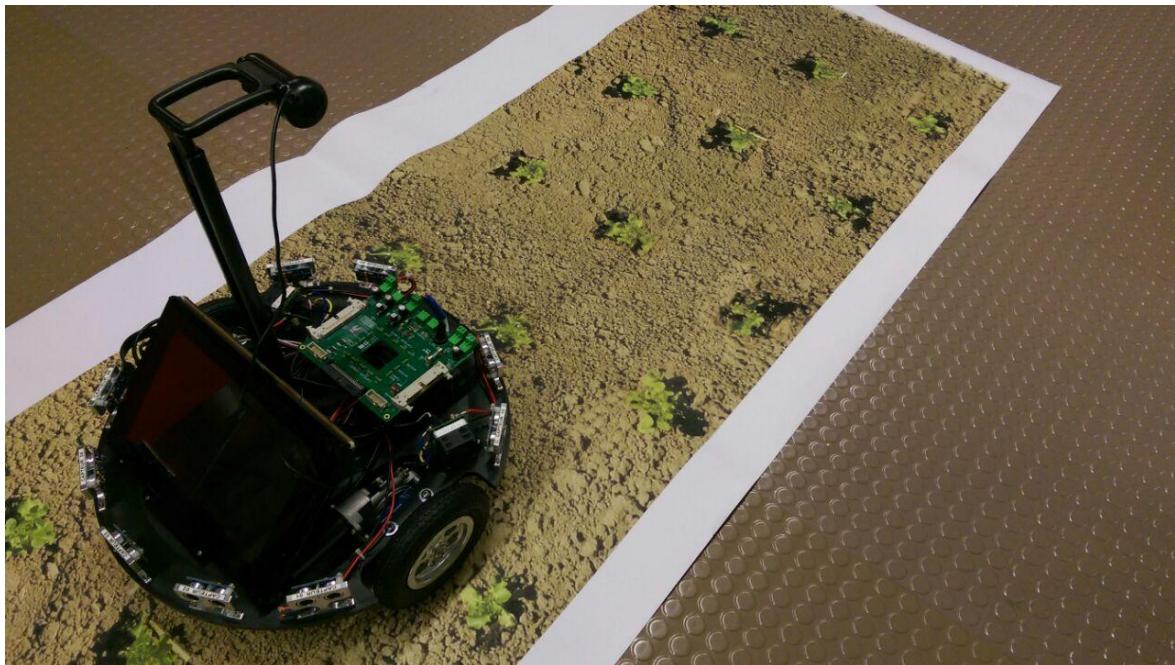


FIGURE 7.3 – Banc de test

Lors des 1ère semaines du projet, une visite a été effectué dans le champ du maraîcher. Lors de cette visite une série de photo et de vidéos ont été prises avec une tablette au sommet d'une brouette. Une fois de retour en laboratoire, les vidéos ont utilisées pour tester et calibrer l'algorithme de détection (voir chapitre 4). Les photos quand à elles, ont été mixées pour créer une grande image de planton de salade qui a été imprimée sur une grande feuille. Le prototype peut s'y promener et il est ainsi possible de tester le pilote automatique et l'arrosage des plantons avec le bras. Les couleurs ressortent quand même beaucoup moins bien sur le papier qu'en réalité. Nous pouvons donc imaginer que les tests en plein jour devraient donner au minimum un résultat équivalent.

Chapitre 8

Conclusion

8.1 Synthèse

Nous voilà arrivé au terme de ce travail qui fût très intéressant mais aussi très intense ! Les spécifications minimales du robot ne sont malheureusement pas atteintes puisqu'aucune vraie version du robot n'a pu être faite actuellement par la partie mécanique. et que d'autres travaux ont dû être effectué en lieu et place de ce qui était prévus. Néanmoins, les points suivants sont terminés et ont pu être testés :

- La détection des plantons fonctionne parfaitement, que ce soit avec la webcam du prototype ou avec une vidéo du monde extérieur
- Le prototype de test a été réalisé et est fonctionnel
- La commande manuelle des roues du prototype fonctionne à merveille
- Il en va de même pour la commande manuelle du bras d'arrosage
- Le pilote automatique peut être activer ou désactivé

De plus, les points suivants seront terminés dans les jours à venir :

- Le suivi de la ligne à vitesse lente et constante en démarrant déjà bien en face
- La simulation de l'arrosage grâce à la commande des bras et à l'allumage de la led situé à l'avant du prototype

Le projet a pris pas mal de retard par rapport à ce qui était prévu initialement. Les raisons principales de ce retard sont les suivantes :

- Durant la 1ère partie de ce travail de bachelor, même si le temps de travail a pu ensuite être rattrapé, le dernier semestre de l'année a été très chargé et il a malheureusement été impossible d'allouer autant de temps que prévu au projet
- Beaucoup plus d'options que prévus ont été codées, principalement pour le programme de détection des plantons. Elles rendent la suite du projet beaucoup plus faciles mais la superposition de ces nombreuses options dans le même programme ont pas contre apporté leur lot de petits bugs. Ceux petits bugs n'étaient pas vraiment compliqués à résoudre mais il y en a eu beaucoup.
- Un problème de fuite mémoire (memory leak) a pris plus de 3 jours à être résolu. Il était dû à une façon d'utiliser une fonction d'OpenCV qui ne respectait pas une close obscure de la documentation de la bibliothèque.
- La création du prototype n'était pas initialement prévue dans le planning et a aussi pris pas mal de temps
- Un dernier point qui a pris plus de temps que prévu a été l'assemblage de toutes les différentes parties du programme. Cela a pris du temps car il a fallut gérer les

problèmes de concurrence mais surtout car il a fallut retravailler une bonne partie du code pour que l'ensemble soit conséquent et propre.

8.2 Futur du projet

Voilà un résumé des informations présente en fin de chapitre. Les prochaines étape du projet, par ordre de priorité sont les suivantes :

1. Se procurer un moteur pas à pas et un driver au plus proche des produits du robot final et adapter le soft pour les faire fonctionner
2. Tester la qualité des caméras en extérieur et se décider à passer à une webcam de meilleure qualité
3. Il serait vraiment bienvenue de migrer le projet vers le langage C++ en lieu et place du C.
4. Coder une petite interface graphique (Front-end) peut rendre les tests encore plus pratique, surtout les tests sur le terrain.
5. Une fois le vrai robot terminé, il sera intéressant de pouvoir tester le fonctionnement de toute ce qui a été conçu sur le terrain, le vrai. Les corrections nécessaires devront ensuite être réalisées.
6. Dans le cas où les performances de la raspberry PI se montraient insuffisantes sur le terrain, il faudrait commander et porter le projet sur une board plus puissante
7. Une fois le projet actuel fonctionnant sur le robot réel, il faudra rendre ce robot autonome et pour ça, un puces GPS et une puce GSM sont indispensables.

8.3 Commentaires personnels

Même si ce projet et ce rapport m'ont bien fait transpirer dans les derniers jours avant le rendu, j'ai pris beaucoup de plaisir tout au long de ce travail! Je suis particulièrement fière de mes différentes initiatives et organisation qui ont permis de coordonner plusieurs personnes pour réaliser le prototype. Par contre, même si j'ai pourtant essayé de le faire tout au long du projet, une communication plus clair et plus régulière avec mon collègue de mécanique aurait permis de mieux anticipé certains changements.

Je remercie mon professeur Etienne Messerli pour sa communication clair, sa souplesse et sa bonne humeur tout au long de du projet. Je remercie également tous les intervenants qui ont contribué à la conception du prototype de test.

Je me réjouis de pouvoir terminer les derniers réglages pour pouvoir avoir la satisfaction de voir ce prototype fonctionner.

Bibliographie

- [1] Quentin RABALL. Image 3d de la conception mécanique du robot finale, 2018.
- [2] Farmbot. Farbot, site officiel. <https://farm.bot/>, 2018.
- [3] Wikipedia. Comparison of single-board computers. https://en.wikipedia.org/wiki/Comparison_of_single-board_computers#CPU,_GPU,_memory, 2018.
- [4] BeagleBoard. Beaglebone® blue. <https://beagleboard.org/blue>, 2017.
- [5] Raspberry PI. Raspberry pi modèle 3b+. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>, 2018.
- [6] Banana PI. Banana pi m3. <http://www.banana-pi.org/m3.html>, 2016.
- [7] castman. Article banana pi <https://www.raspberrypi.org/downloads/noobs/> m3 coup de gueule. <https://castman.fr/wordpress/mes-coups-de-gueule/banana-pi-m3-cest-un-vrai-desastre-instable-et-non-finalise-cest-toute-la-conception-qui-est-a-revoir-mais-jai-des-solutions-pour-vous-aider/>, 2016.
- [8] ASUS. Asus tinker board s. <https://www.asus.com/ch-fr/Single-Board-Computer/Tinker-Board-S/>, 2010.
- [9] BeagleBoard. Image de la beaglebone blue. https://cdn-images-1.medium.com/max/1600/1*L_DzTBWW6u5p6VUFLoY8zQ.jpeg, 2017.
- [10] Sparkfun. Image de la raspberry 3 modèle b+. https://cdn.sparkfun.com/assets/parts/1/2/8/2/8/14643-Raspberry_Pi_3_B_-05.jpg, 2018.
- [11] Banggood. Image de la banana pi m3. <https://www.banggood.com/fr/BPI-M3-8-Core-Banana-PI-BPI-M3-Module-with-WIFI-Bluetooth-SATA-Wntenna-p-1030478.html>, 2016.
- [12] LDLC. Image de l'asus tinker board s. https://media.ldlc.com/ld/products/00/04/49/66/LD0004496688_2.jpg, 2017.
- [13] board db.org. Comparaison de la beaglebone blue, de la raspberry pi 3b+, de la banana pi m3 et de la tinker boasrd s d'asus. <https://www.board-db.org/compare/180,251,115,226/>, 2018.
- [14] Raspberry PI. Os noobs installers. <https://www.raspberrypi.org/downloads/noobs/>, 2018.
- [15] Lya (GeckoGeek.fr). Tutorial opencv : Isoler et traquer une couleur. <http://www.geckogeek.fr/tutorial-opencv-isoler-et-traquer-une-couleur.html>, 2010.

- [16] Logitech. Logitech hd pro webcam c920. <https://www.logitech.com/fr-fr/product/hd-pro-webcam-c920>, 2012.
- [17] Hafiz Muhammad Ali pour "Omnicore Agency". 10 best webcams and conference camera for video interviews & streaming in 2018. <https://www.omnicoreagency.com/best-webcams-and-conference-cameras/>, 2018.
- [18] Nico pour "Chromebookeur". Les 6 (vraies) meilleures webcams de 2018 – comparatif complet. <https://chromebookeur.com/meilleures-webcams/>, 2018.
- [19] Nicolas pour "Le Juste Choix". Top 3 des meilleures webcams : comparatif 2018. <https://lejustechoix.fr/meilleurs-webcams-comparatif/>, 2018.
- [20] Uctrlonics. Arducam csi to hdmi cable extension module with 15pin 80mm fpc cable for raspberry pi camera specific (pack of 2).
- [21] Logitech. Image de la webcam hd pro c920 de chez logitech. <https://assets.logitech.com/assets/54515/hd-webcam-pro-c920-gallery.png>, 2012.
- [22] Logitech. Image de la webcam quickcam pro 9000 de chez logitech. <https://secure.logitech.com/assets/18745/18745.png>, 2007.
- [23] Raspberry PI. Image de la camera module v2 de chez raspberry pi. <https://www.raspberrypi.org/app/uploads/2017/05/Pi-Camera-front-1-462x322.jpg>, 2016.
- [24] Raspberry PI. Image de la camera module v2 black de chez raspberry pi. <https://www.raspberrypi.org/app/uploads/2017/05/Pi-Camera-NoIR-front-462x322.jpg>, 2016.
- [25] Elecrow. Elecrow 11.6 inch 1920x1080 hdmi 1080p led display for raspberry pi. <https://www.elecrow.com/elecrow-11-6-inch-1920x1080-hdmi-1080p-led-display-for-raspberry-pi.html>, 2016.
- [26] Banana PI. Écran tactile 9 pouces 1024 x 600 pixels pour le banana pi et compatible raspberry pi. <https://e.banana-pi.fr/afficheurs-ecrans/288-ecran-tactile-9-pouces-1024-x-600-pixels-pour-le-banana-pi-et-compatible-raspberry-pi.html>, 2016.
- [27] Waveshare. 10.1inch hdmi lcd (b) (with case). [https://www.waveshare.com/wiki/10.1inch_HDMI_LCD_\(B\)_\(_with_case\)_](https://www.waveshare.com/wiki/10.1inch_HDMI_LCD_(B)_(_with_case)_), 2016.
- [28] Elecrow. Image de l'elecrow 11.6 inch 1920x1080 hdmi 1080p led display for raspberry pi. https://www.elecrow.com/media/catalog/product/cache/1/small_image/9df78eab33525d08d6e5fb8d27136e95/e/1/elecrow_11.6_inch_1920x1080_hdmi_1080p_led_display_for_raspberry_pi_2.jpg, 2016.
- [29] Waveshare. Image de l'écran 10.1inch hdmi lcd (b) (with case). https://www.waveshare.com/media/catalog/product/cache/1/image/800x800/9df78eab33525d08d6e5fb8d27136e95/1/0/10.1inch-hdmi-lcd-b-with-holder-front_1.jpg, 2016.

- [30] Banana PI. Image de l'Écran tactile 9 pouces 1024 x 600 pixels pour le banana pi et compatible raspberry pi. <https://e.banana-pi.fr/afficheurs-ecrans/288-ecran-tactile-9-pouces-1024-x-600-pixels-pour-le-banana-pi-et-compatible-raspberry-pi.html>, 2016.
- [31] Sylvie Ladet² Renaud Lahaye¹. Les principes du positionnement par satellite :gnss . 1. Institut de Développement de la Géomatique (IDGEO); 42, avenue du Général de Croutte, F-31100 Toulouse, France ; renaud.lahaye@idgeo.fr
2. UMR 1201 DYNAFOR, INRA; 24 Chemin de Borde Rouge- Auzerville, CS 52627, F-31326 Castanet -Tolosan cedex; France
https://www6.inra.fr/cahier_des_techniques/.../2/.../05_CH1_LAHAYE_gnss.pdf, 2016.
- [32] Embedded Artists. Image de la gps receiver board. https://www.embeddedartists.com/products/acc/acc_gps.php, 2017.
- [33] Cyril Masson. Connectivité dans l'iot : la carte sim a encore de l'avenir! <https://aruco.com/2017/02/connectivite-iot-carte-sim-cyril-masson/>, 2017.
- [34] matooma. Carte sim iot. <https://www.matooma.com/fr/produits-et-services/cartes-sim-m2m>, 2018.
- [35] Cooking Hacks by Libelium. Gprs/gsm quadband module for arduino and raspberry pi tutorial (sim900). <https://www.cooking-hacks.com/documentation/tutorials/gprs-gsm-quadband-module-arduino-raspberry-pi-tutorial-sim-900/#links>, 2017.
- [36] Adafruit. 16-channel 12-bit pwm/servo driver - i2c interface. <https://www.adafruit.com/product/815>, 2015.
- [37] Wikipedia. Description du format hsv. https://fr.wikipedia.org/wiki/Teinte_Saturation_Valeur, 2018.
- [38] Wikipedia. Cercle chromatique avec graduation de l'angle pour définir la teinte. https://commons.wikimedia.org/wiki/File:CYM_color_wheel.png, 2010.
- [39] Christian RONSE. Opérations morphologiques de base : dilatation, érosion, ouverture et fermeture binaires. <https://dpt-info.u-strasbg.fr/~cronse/TIDOC/MM/deof.html>, 2013.
- [40] Ragnar Hojland Espinosa. Joystick api documentation. <https://www.kernel.org/doc/Documentation/input/joystick-api.txt>, 1998.
- [41] Jason White. Reads joystick/gamepad events on linux and displays them. <https://gist.github.com/jasonwhite/c5b2048c15993d285130>, 1998.

Chapitre 9

Authentication

Par la présente, je soussigné, Ludovic Richard, déclare avoir réalisé seul ce travail et ne pas avoir utilisé d'autres sources que celles citées dans la bibliographie.

Date

Ludovic Richard

Signature

Table des figures

2.1	Schémas du champs[1]	4
2.2	Concept mécanique du robot[1]	5
3.1	BeagleBone Blue[9]	14
3.2	Raspberry PI modèle 3B+[10]	14
3.3	Banana PI M3[11]	14
3.4	Asus Tinker Board S[12]	14
3.5	Webcam Logitech Pro C920[21]	16
3.6	Logitech Quickcame Pro 9000[22]	16
3.7	Raspberry PI Camera Module V2[23]	16
3.8	Raspberry PI Camera Module V2 Black[24]	16
3.9	Raspberry Camera Module V2	17
3.10	Raspberry Camera Module V2 black	17
3.11	Webcam HD Pro 9700	17
3.12	Raspberry Camera Module V2	17
3.13	Raspberry Camera Module V2 black	17
3.14	Raspberry Camera Module V2	18
3.15	Raspberry Camera Module V2 black	18
3.16	Webcam HD Pro 9700	18
3.17	Raspberry Camera Module V2	18
3.18	Raspberry Camera Module V2 black	18
3.19	Écran Elecrow 11,6'[28]	20
3.20	Écran tactile Waveshare 10,1'[29]	20
3.21	Écran tactile Banana PI 9'[30]	20
3.22	GPS RECEIVER BOARD[32]	21
3.23	GPRS/GSM Quadband Module for Arduino and Raspberry Pi Tutorial (SIM900)[35]	22
3.24	Carte d'extension PWM de chez Adafruit[36]	23
4.1	Exemple d'une binarisation sur une image de 4x4 pixels	27
4.2	Cercle chromatique[38]	28
4.3	Exemple de réglage - Fenêtre binarisation	32
4.4	Reconnaissance des plantons test 1 - Image source	34
4.5	Reconnaissance des plantons test 1 - Binarisation	35
4.6	Reconnaissance des plantons test 1 - Érosion	35
4.7	Reconnaissance des plantons test 1 - Dilatation	36
4.8	Reconnaissance des plantons test 1 - Séparation en objets distincts	36
4.9	Reconnaissance des plantons test 1 - Ajout des marqueurs sur l'image source	37
4.10	Reconnaissance des plantons test 2 - Image source	38
4.11	Reconnaissance des plantons test 2 - Binarisation	39
4.12	Reconnaissance des plantons test 2 - Érosion	39

4.13 Reconnaissance des plantons test 2 - Dilatation	40
4.14 Reconnaissance des plantons test 2 - Séparation en objets distincts	40
4.15 Reconnaissance des plantons test 2 - Ajout des marqueurs sur l'image source	41
5.1 Gestion des roues - Mode n°0	46
5.2 Gestion des roues - Mode n°1	47
5.3 Exemple de détection de ligne validée	48
5.4 Exemple de détection de ligne refusée	48
6.1 Schéma complet de la structure du programme	51
7.1 Prototype	56
7.2 Prototype de bras d'arrosage	56
7.3 Banc de test	57