



**HAL**  
open science

# Optimisation d'un système robotisé de perception active pour la détection de maladies sur des feuilles de betteraves sucrières

R. Laborde

► **To cite this version:**

R. Laborde. Optimisation d'un système robotisé de perception active pour la détection de maladies sur des feuilles de betteraves sucrières. Sciences de l'environnement. 2019. hal-02609826

**HAL Id: hal-02609826**

**<https://hal.inrae.fr/hal-02609826v1>**

Submitted on 16 May 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# POLYTECH Montpellier

## Stage industriel de fin d'études Département MEA

*ANNEE 2018 - 2019*

Optimisation d'un système robotisé  
de perception active pour la détection  
de maladies sur des feuilles de  
betteraves sucrières

Rémi Laborde

STAGE



**MEA**



ECOLE POLYTECH Montpellier  
UNIVERSITE DE MONTPELLIER - Campus Triolet  
Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5  
Tél. : 04 67 14 31 60 - Fax : 04 67 14 45 14  
E-mail : scola@polytech.univ-montp2.fr



# Remerciements

*Mes remerciements iront à tous ceux qui m'ont assisté durant ce projet et contribué à son bon déroulement :*

*Mr Bernard BENET, ingénieur/chercheur pour son encadrement et ses conseils durant toute la durée du stage.*

*Mme Fabienne MAUPAS, Responsable du projet Phenaufol,*

*Mr Roland LENAIN, Chef de l'équipe ROMEA,*

*Mr Jean-Pierre CHANET, Chef de l'unité de recherche TSCF,*

*Mr Emmanuel HUGO, Directeur régional d'IRSTEA de Clermont-Ferrand,*

*Mr Vincent ROUSSEAU, responsable informatique, pour son aide précieuse à la mise en place de l'environnement de travail et ses conseils pour l'utilisation de ROS et GitLab,*

*Mr André CROSNIER, mon tuteur à Polytech Montpellier, pour le suivi effectué.*

---

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Présentation de l’Institut Technique de la Betterave (ITB)</b>	<b>2</b>
<b>3</b>	<b>Présentation d’IRSTEA</b>	<b>3</b>
3.1	Historique . . . . .	3
3.2	L’unité de recherche TSCF (Technologies et Systèmes de Clermont-Ferrand) . . .	3
3.3	L’équipe ROMEA . . . . .	3
<b>4</b>	<b>Enjeux du Développement Durable et Responsabilité Sociétale (DDRS)</b>	<b>4</b>
<b>5</b>	<b>Le projet Phenaufol</b>	<b>5</b>
5.1	Présentation du projet . . . . .	5
5.2	Historique . . . . .	5
5.3	Contexte . . . . .	5
5.4	Objectifs du projet . . . . .	6
5.5	Contraintes . . . . .	6
5.5.1	Contraintes environnementales . . . . .	6
5.5.2	Contraintes matérielles . . . . .	6
<b>6</b>	<b>Environnement de travail</b>	<b>7</b>
6.1	Environnement matériel . . . . .	7
6.1.1	Le robot UR5 . . . . .	7
6.1.2	La caméra Baumer . . . . .	7
6.2	Environnement logiciel . . . . .	8
6.2.1	Présentation du système d’exploitation ROS . . . . .	8
6.2.2	Outil de Deep Learning Tensorflow . . . . .	9
6.2.3	Librairie de traitement d’images OpenCV . . . . .	9
6.2.4	Gestionnaire de version Gitlab . . . . .	9
<b>7</b>	<b>Travail existant</b>	<b>10</b>
7.1	Détection des centres des plants de betteraves . . . . .	10
7.1.1	Présentation de l’algorithme . . . . .	10
7.1.2	Utilisation du réseau de neurones Inception V3 . . . . .	12
7.1.3	Critique et analyse de l’algorithme existant . . . . .	12
7.2	Segmentation des feuilles de betteraves . . . . .	14
7.2.1	Présentation de l’algorithme . . . . .	14
7.2.2	Critique et analyse de l’algorithme existant . . . . .	16
7.3	Contrôle/Commande du robot . . . . .	17
7.3.1	Fonctionnement général de l’algorithme de commande . . . . .	17
7.3.2	Description de l’algorithme de détection du cylindre . . . . .	17
7.3.3	Description de l’algorithme de détection lié à la commande . . . . .	18
<b>8</b>	<b>Amélioration des programmes existants</b>	<b>19</b>
8.1	Détection des centres des plants de betteraves . . . . .	19
8.2	Segmentation des feuilles de betteraves . . . . .	19

<b>9</b>	<b>Recherches d'un nouveau algorithme pour la segmentation de feuilles</b>	<b>19</b>
9.1	Recherches bibliographiques . . . . .	19
9.2	Implémentation du réseau Mask R-CNN . . . . .	22
9.2.1	Présentation du réseau . . . . .	22
9.2.2	Entraînement du réseau . . . . .	23
9.2.3	Tests et résultats . . . . .	24
<b>10</b>	<b>Recherches d'un algorithme pour le suivi d'objets dans une image</b>	<b>26</b>
10.1	Recherches bibliographiques . . . . .	26
10.2	Tests de différentes solutions pour le tracking . . . . .	28
<b>11</b>	<b>Conclusion</b>	<b>30</b>
<b>12</b>	<b>Travail réalisé après la soutenance de stage</b>	<b>31</b>
12.1	Suite des tests sur le tracking . . . . .	31
12.2	Modifications de l'algorithme de détection lié à la commande . . . . .	32
12.3	Plante artificielle pour la simulation et entraînement du réseau . . . . .	34
12.4	Machine à états du tracking . . . . .	35
12.5	Travail restant . . . . .	35
<b>13</b>	<b>Annexes</b>	<b>36</b>
13.1	Bibliographie . . . . .	36
13.1.1	Bibliographie pour la segmentation/détection des feuilles de betteraves . . . . .	36
13.1.2	Bibliographie pour le tracking . . . . .	39
13.2	Principales fonctions d'OpenCV utilisées . . . . .	41
13.3	Explication du calcul de la précision moyenne de la partie 9.2.3 . . . . .	43
13.4	Architecture logicielle du projet ROS . . . . .	45
13.4.1	Architecture des packages ROS . . . . .	45
13.4.2	Librairies complémentaires . . . . .	46
13.4.3	Fichiers/Dossiers complémentaires . . . . .	46
13.5	Architecture du dossier Mask R-CNN . . . . .	47
13.6	Tutoriel pour réaliser l'entraînement du réseau sur de nouvelles images . . . . .	48
13.7	Déroulement du projet . . . . .	49
13.7.1	Diagramme de Gant . . . . .	49
13.7.2	Problèmes rencontrés . . . . .	50
13.8	Liste des figures . . . . .	51

# 1 Introduction

Ce stage de fin d'études s'inscrit dans le cadre du projet Casdar Phenaufol qui concerne le développement d'un système robotisé embarquant un capteur de perception de type caméra couleur destiné à l'acquisition et au traitement d'images sur des parcelles de betteraves sucrières, pour détecter la présence de maladies sur des feuilles de betteraves et étudier la propagation de celles-ci dans différentes cultures et à différents stades de croissance.

L'objectif de ce stage consiste donc à optimiser le système de perception active réalisé sous ROS comprenant 3 parties complémentaires :

- la détection des centres de betteraves par traitement d'images,
- la détection des feuilles de betteraves par traitement d'images,
- le contrôle/commande du robot pour positionner la caméra à l'endroit désiré au-dessus de chaque feuille.

Dans un premier temps, je m'attacherai à présenter le projet et l'environnement de travail. Dans un second temps, j'analyserai l'existant aussi bien pour la partie perception que pour la partie commande du système robotisé.

Ensuite, je présenterai le travail réalisé, les différentes recherches effectuées et leurs mises en oeuvre. Enfin, je ferai une synthèse du travail accompli, tout en proposant des améliorations possibles. Pour terminer, j'énoncerai les tâches restantes<sup>1</sup> à finaliser d'ici la fin de ce stage.

---

1. Pour plus d'informations, le déroulement du projet est détaillé en [annexe](#).

## 2 Présentation de l'Institut Technique de la Betterave (ITB)

L'Institut Technique de la Betterave (ITB) a été créé en 1944 par la Confédération des Planteurs de Betteraves (CGB) et le Syndicat National des Fabricants de Sucre (SNFS).

L'ITB s'inscrit dans les attentes sociétales et environnementales et mène des études dans 4 grands thèmes :

### **Génétiques et variétés**

- contribuer aux orientations génétiques des sélectionneurs,
- évaluer les variétés et leur durabilité,
- conseiller les agriculteurs dans leurs choix variétaux

### **Désherbage, maladies et ravageurs**

- tester et proposer des programmes efficaces, économiques et respectueux de l'environnement,
- participer aux réseaux de surveillance et alerter,
- mettre en œuvre une protection intégrée.

### **Agronomie**

- évaluer de nouvelles pratiques culturales,
- ajuster l'irrigation et la fertilisation,
- raisonner la betterave dans la rotation.

### **Agroéquipements**

- tester les innovations technologiques,
- conseiller les réglages appropriés à chaque matériel.

Ces missions impliquent la mise en place d'actions et de projets qui regroupent les experts nationaux de l'ITB et ses délégués régionaux avec de nombreux partenaires dont la Recherche publique, les Chambres d'Agriculture, les partenaires semenciers, les constructeurs de machines, les firmes phytopharmaceutiques. L'ITB est membre de l'IIRB (Institut International de Recherches Betteravières) qui regroupe notamment les instituts techniques européens.

L'ITB est membre de nombreux réseaux de recherche et développement, ce qui permet à l'institut d'être au cœur des réflexions scientifiques et techniques sur les grandes cultures, tant au niveau national qu'international.

Les performances scientifiques et techniques de l'ITB s'appuient sur un parfait équilibre entre recherche appliquée, présence et expérimentation terrain.

L'ITB, avec ses 8 délégations régionales, est implanté au plus près des producteurs de betteraves. Plus de la moitié de ses 38 collaborateurs sont situés dans les régions betteravières. Ses délégués régionaux constituent une interface essentielle entre les chercheurs et le terrain.

## 3 Présentation d'IRSTEA

IRSTEA (Institut National de Recherche en Sciences et Technologies pour l'environnement et l'Agriculture) est un établissement public à caractère scientifique et technologique (EPST). Réparties en France dans 9 centres, ses activités de recherche et d'expertise sont tournées vers l'action et l'appui aux politiques publiques. Ses principaux domaines de recherche concernent les enjeux majeurs d'une agriculture responsable et de l'aménagement durable des territoires, la gestion de l'eau et les risques associés, sécheresse, crues, inondations, l'étude des écosystèmes complexes et de la biodiversité dans leurs interrelations avec les activités humaines.

### 3.1 Historique

IRSTEA est né de la fusion de deux centres publics de recherche, le CTGREF (Centre Technique du Génie Rural et des Eaux et Forêts) et le CNEEMA (Centre National d'Etude et d'Expérimentation de Machinisme Agricole) en 1981. Il s'appelait alors Cemagref (Centre national du machinisme agricole, du génie rural et des eaux et forêts) mais a été renommé en 2012 pour mieux coïncider avec la réalité des recherches effectuées, qui ont évolué en 30 ans. Au premier janvier 2020, il est prévu de fusionner l'IRSTEA avec l'INRA (Institut National de la Recherche Agronomique) car ils ont des thèmes de recherches proches.

### 3.2 L'unité de recherche TSCF (Technologies et Systèmes de Clermont-Ferrand)

L'UR TSCF est composée de 3 équipes de recherche qui rassemblent 60 agents :

- Plateau de Recherche Technologique Pôle Épandage Environnement (PRT PEE)
- RObotique et Mobilité pour l'Environnement et l'Agriculture (ROMEA)
- Systèmes d'information agri-environnementaux communicants (COPAIN)

Elle mobilise les sciences pour l'ingénieur, les technologies de l'information et de la communication pour conduire des recherches sur les méthodes et outils pour une ingénierie des systèmes agro-environnementaux. Elle conduit également des activités de recherche, d'expertise et d'essai dans le domaine de la sécurité et des performances des agroéquipements pour contribuer à l'amélioration de la sécurité en agriculture et à la réduction des pollutions d'origine agricole.

### 3.3 L'équipe ROMEA

Au sein de l'unité de recherche TSCF, l'équipe ROMEA, basée à Clermont-Ferrand (63), conçoit des systèmes reconfigurables et à autonomie partagée, pour accroître les performances et la sécurité des engins œuvrant en milieux naturels, en particulier ceux rencontrés dans l'agriculture. Cette équipe concentre ses travaux sur trois axes :

- La conception de dispositifs de sécurité, aussi bien du point de vue de la conception mécanique que de la réalisation de dispositifs d'assistance.
- La perception de l'environnement et la modélisation du comportement d'un robot et de son interaction avec l'utilisateur.
- Le développement d'algorithmes avancés de commandes ou d'assistance, en prenant en compte les phénomènes perturbants et les incertitudes liées à l'évolution en milieu tout-terrain et dans différentes conditions.

## 4 Enjeux du Développement Durable et Responsabilité Sociétale (DDRS)

### Quelle est la position de l'entreprise vis à vis des enjeux du DDRS ?

La filière doit répondre aux attentes des pouvoirs publics, de la société c'est-à-dire des consommateurs de plus en plus sensibilisés à la qualité sanitaire des produits et aux impacts environnementaux des modes de production. Les préoccupations environnementales et la gestion des ressources naturelles seront pleinement intégrées dans les travaux de l'ITB : adaptation aux changements climatiques même si ses effets sont pour le moment perçus favorables à la productivité de la betterave, poursuite de la réduction des intrants (fertilisants, phytosanitaires dans le contexte du programme Ecophyto II), réduction des émissions de polluants atmosphériques, diminution des pratiques recourant à la consommation d'énergie fossile et gestion de la ressource en eau.

### Quels sont les leviers qui motivent l'intérêt de l'entreprise pour le DDRS ?

- La nécessité de maintenir une filière betteravière française économiquement forte et pérenne.
- Une forte attente de la société pour une évolution de l'agriculture vers des systèmes plus durables. Cette attente se traduit au niveau de l'état par différents plans d'actions vers une agriculture agro-écologique et un dynamisme des territoires agricoles.
- Une évolution des modes de consommation qui impacte les orientations de l'agriculture d'aujourd'hui. La forte croissance de l'agriculture biologique avec 20 % d'augmentation par rapport à 2015.

### Quelle est l'organisation mise en place pour répondre à ces questions ?

L'ITB aura un rôle central dans l'anticipation et l'accompagnement de la filière dans le développement d'une filière biologique qui nécessitera de nombreuses innovations.

L'ITB doit donc anticiper et accompagner la filière dans des propositions techniques, et dans l'évaluation d'une filière actuellement en gestation. L'investigation technique sur cette thématique ne sera possible qu'avec une prise en compte des contraintes liées au volet industriel, aux périodes de récolte et à la durée de campagne. La mise en place d'une telle filière demandera donc un engagement et une concertation de tous ses acteurs. L'ITB aura un rôle de coordination de l'ensemble des actions entreprises pour mener à bien ce projet.

### Quelles sont les actions mises en œuvre ?

Quelques exemples de projets mis en œuvre :

#### Gestion des maladies fongiques foliaires

L'objectif est de développer et de promouvoir un mode de gestion des risques de maladies foliaires, et de déclenchement des interventions fongicides, basés sur le contexte de la parcelle et sur le choix variétal.

#### Recherche d'alternatives aux néonicotinoïdes (pesticides nocifs pour les abeilles)

L'arrêt programmé de cette protection phytosanitaire, qui pourrait avoir des conséquences importantes en termes de rendement, impose à la filière betteravière de se tourner vers des solutions alternatives efficaces. Pour ce faire, et compte tenu des similitudes entre les différents pathosystèmes observés sur betterave, céréales et colza, l'ITB a décidé de coordonner ses efforts avec les deux autres instituts techniques des grandes cultures (ARVALIS - Institut du végétal et Terres Inovia) ainsi qu'avec une équipe de virologie de l'INRA et Bayer CropScience.

## 5 Le projet Phenaufol

### 5.1 Présentation du projet

Le projet Casdar Phenaufol 2017-2020, porté par l'ITB (Institut Technique de la Betterave) en collaboration avec IRSTEA (Equipe ROMEA - UR TSCF de Clermont-Ferrand) et l'UMR Agroécologie de Dijon, a pour objectif d'automatiser la détection, l'identification des maladies foliaires de la betterave.

Le but recherché est d'automatiser la détection, l'identification et la quantification des maladies foliaires (oïdium, cercosporiose) de la betterave grâce à la réalisation d'un système autonome complet comprenant des capteurs de diverses natures notamment de type caméra pour aider les agriculteurs à limiter l'utilisation de pesticides pendant les différents stades de croissance des plantes.

Dans le cadre des travaux d'expérimentation de l'ITB, cela permettrait de développer d'avantage l'usage de variétés résistantes aux maladies foliaires et d'entrevoir de nouvelles pratiques agricoles plus efficaces et plus respectueuses de l'environnement.

### 5.2 Historique

Le centre IRSTEA de Clermont-Ferrand a participé à différents projets de développement, avec l'ITB (Institut Technique de la Betterave), avec pour objectif le développement de systèmes automatisés pour effectuer des acquisitions d'images couleurs sur les parcelles de betteraves sucrières.

Ces projets permettaient de réaliser des mesures telles que le comptage des plants, la mesure de l'espacement entre les plants sur les rangées et des mesures de surfaces foliaires, pour ne plus utiliser des méthodes manuelles très contraignantes et fastidieuses. Les mesures réalisées, permettent à l'ITB de comparer diverses variétés de betteraves, sur les mêmes parcelles et à différents stades de croissance des plants, du premier stade de croissance jusqu'au stade de final.

Les deux principaux systèmes d'acquisition d'images nommés BECAM (BEet Count Automatic Machine) ont été développés par le centre IRSTEA de Clermont-Ferrand. En 2003, un système guidé manuellement par un opérateur a été mis en œuvre. Trois rangées de betteraves étaient traitées en parallèle. En 2009, un système de type portique avec chariot motorisé a été réalisé.

### 5.3 Contexte

En 2015, l'ITB a mené les premières discussions avec IRSTEA, visant la robotisation d'un système de mesure sur les rangées de betteraves, en travaillant sur deux nouveaux points : détection des maladies sur les plants et l'étude du phénotypage<sup>2</sup> des plants.

En 2016, l'ITB a ainsi proposé à IRSTEA de réaliser une étude de faisabilité d'un système robotisé comportant une plateforme mobile avec divers types de capteurs embarqués : il s'agissait d'effectuer des acquisitions de données sur le terrain et de réaliser des mesures en temps réel et en temps différé avec des méthodes de traitement de données complexes. Depuis janvier 2017, l'ITB est leader du projet Phenaufol, lauréat de l'appel à projet Casdar « Recherche Technologique » en 2016.

---

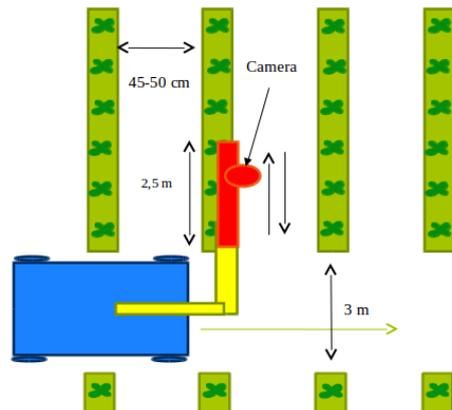
2. Le phénotypage végétal est l'identification des caractéristiques observables d'un lot de plantes et son suivi lors de la croissance.

## 5.4 Objectifs du projet

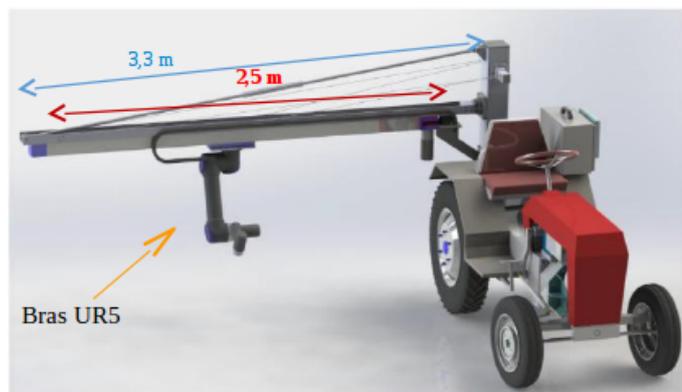
Le travail de perception active d'IRSTEA consiste à développer des algorithmes de traitement d'images, permettant d'une part, d'identifier et séparer différents plants de betteraves sur une rangée, et d'autre part, de détecter et discriminer les différentes feuilles pour chacun d'eux.

Parallèlement à ces informations de vision artificielle obtenues en temps réel, des algorithmes de contrôle/commande des 7 degrés de liberté du système robotisé sont appliqués pour positionner l'extrémité du bras porteur de la caméra dans des positions et orientations désirées au-dessus des feuilles de betteraves sucrières de façon à acquérir des images pour différentes feuilles.

Sur ces images, des méthodes de traitement d'images développées par l'UMR agroécologie de Dijon seront appliquées pour détecter des maladies sur les feuilles de betteraves (maladie de type oïdium et cercosporiose).



(a) Navigation du robot dans les parcelles de betteraves



(b) Représentation du système complet sur un tracteur

FIGURE 1 – Représentations du robot Phenaufol

## 5.5 Contraintes

### 5.5.1 Contraintes environnementales

Concernant les algorithmes de détection développés à IRSTEA, plusieurs contraintes sont à prendre en compte pour la détection des plants et feuilles de betteraves.

Les programmes de détection doivent faire abstraction de la luminosité extérieure dans laquelle se trouve le robot. En effet, les expérimentations sont effectuées de jour et sont donc sujettes aux conditions météorologiques (temps ensoleillé, nuageux, ombre du robot sur les plants de betteraves, etc...).

De plus, les photos étant prises à différents stades de croissance des plantes, la taille et la forme de celles-ci ne doivent pas influencer les résultats de détection, que ce soit pour la reconnaissance d'un plant ou de feuilles de betteraves.

### 5.5.2 Contraintes matérielles

De part la structure du robot, on peut noter plusieurs inconvénients :

- Le système complet peut être très lourd et très encombrant,
- Il n'y a qu'un seul capteur mobile situé à l'extrémité du bras,
- La longueur du bras étant limitée, le capteur ne pourra donc pas couvrir la totalité de la longueur des rangées.

## 6 Environnement de travail

### 6.1 Environnement matériel

Afin d'atteindre les objectifs du projet, un robot nommé Phenaufol, développé par la société ROBOTNIK a été utilisé. Ce robot est composé d'un axe linéaire sur lequel coulisse un bras manipulateur UR5, de Universal Robots. Une caméra couleur est située à l'extrémité du bras UR5. Une armoire de commande électrique et un ordinateur central assurent la coordination de l'ensemble de ces éléments. Ce système robotisé a pour fonction de réaliser des acquisitions d'images sur des plants de betteraves. Il comprend 7 degrés de liberté : 6 pour le bras UR5 et le septième pour le déplacement du bras sur l'axe linéaire.

Pour des expérimentations sur le terrain (champ de betteraves sucrières), le robot sera attelé sur un microtracteur durant la campagne 2019.



FIGURE 2 – Photo du robot Phenaufol

#### 6.1.1 Le robot UR5

Le robot UR5 est un robot à six degrés de liberté possédant six axes de rotations. Il peut porter jusqu'à 5 kg et possède la particularité d'être « collaboratif », c'est-à-dire qu'il possède des capteurs de pression intégrés qui lui permettent de s'arrêter automatiquement s'il rencontre un obstacle inattendu.

#### 6.1.2 La caméra Baumer

La caméra utilisée est un modèle Baumer VLG40C. Il s'agit d'une caméra Gigaethernet grand angle avec une résolution native de 2044x2044 pixels. Elle est utilisée en mode « binning » (résolution 1022x1022) pour réduire le temps de traitement des images.

## 6.2 Environnement logiciel

### 6.2.1 Présentation du système d'exploitation ROS

ROS (Robot Operating System) est un système d'exploitation composé d'outils, de bibliothèques visant à réduire la complexité concernant la procédure d'écriture de comportements robotiques complexes et robustes. Dans le cadre du projet, nous utilisons la version ROS Kinetic. L'architecture du projet Phenaufol comportant plusieurs packages ROS sera détaillée en [annexe](#).

#### Les notions de base de ROS

##### Le master

Le Master est un service de déclaration et d'enregistrement des nœuds. Le Master est la base de données dans laquelle les nœuds peuvent stocker de l'information. Sans le master, les nodes ne pourraient pas se connaître entre eux et échanger des informations.

##### Les nodes

Les nodes sont des processus qui effectuent des calculs. Chaque node se déclare au Master. Un système de contrôle de robot comprend généralement plusieurs nodes. Par exemple, il peut y avoir un node pour le contrôle d'un télémètre laser, un node pour le contrôle d'un moteur, un node pour le traitement des données d'un capteur etc...

##### Les topics

Un topic est un système de transport de l'information basé sur le système de l'abonnement / publication. Le topic est un nom utilisé pour identifier le contenu du message. Un node envoie un message en le publiant sur un topic donné. Un node intéressé par un certain type de données s'abonnera au topic approprié. On peut considérer un topic comme un bus de messages. Chaque bus a un nom et n'importe qui peut se connecter au bus pour envoyer ou recevoir des messages, à condition qu'ils soient du bon type.

##### Les messages

Les nodes communiquent entre eux en transmettant des messages. Un message est simplement une structure de données comprenant des champs typés. Les types standards (entiers, flottants, booléens, etc.) sont supportés. Par exemple, un node peut publier sur un topic, un message contenant un tableau de flottants représentant les positions des moteurs du robot.

##### Les services

Le concept de services a été introduit pour répondre au problème des processus synchrones. Ces services sont représentés par une chaîne de caractères et une paire de messages. L'un est responsable de la demande et l'autre de la réponse. D'autre part, les services diffèrent des topics. Ils peuvent travailler avec un seul node et les nodes peuvent envoyer leurs messages à un seul service.

## 6.2.2 Outil de Deep Learning Tensorflow

« TensorFlow » est une bibliothèque open source développée par Google pour le Machine Learning et Deep Learning. Le Deep Learning est beaucoup utilisé en traitement d'images pour faire de la reconnaissance d'objets.

Le réseau de neurones « Inception V3 » créé par Google est utilisé. Cette bibliothèque permet de faire du « transfer learning ». Cela consiste à récupérer un modèle déjà entraîné pour la reconnaissance d'objet dans une image et de fournir la base de données sur laquelle le réseau de neurones s'entraînera.

Pour utiliser « Tensorflow » et les bibliothèques associées, il est nécessaire d'installer les packages suivants : `tensorflow_ros_cpp`, `tensorflow_catkin` et `catkin simple`.

## 6.2.3 Librairie de traitement d'images OpenCV

« OpenCV » est une bibliothèque open source gratuite pour le traitement d'image. Elle donne accès à de nombreuses fonctions pour différentes applications de vision artificielle. Plusieurs fonctions morphologiques de base ainsi que différents algorithmes de la bibliothèque « OpenCV » ont été utilisés.

L'ensemble des fonctions utilisées dans ce projet seront détaillées dans l'analyse de l'existant et en annexe, que ce soit pour la détection des centres des plants de betteraves ou la segmentation des feuilles.

## 6.2.4 Gestionnaire de version Gitlab

« Gitlab » est une plateforme permettant d'héberger et de gérer des projets.

Elle offre la possibilité de gérer les dépôts « Git » et ainsi de mieux appréhender la gestion des versions des codes sources.

## 7 Travail existant

Le travail présenté ici est divisé en trois parties :

La première concerne la détection des centres des plants de betteraves, la seconde porte sur la détection des feuilles et la troisième concerne la partie commande du robot liée à la vision.

Pour les deux algorithmes de traitement d'images (centres et feuilles), je vais détailler pas à pas le fonctionnement de chacun d'eux et pour chaque étape, j'indiquerai les fonctions utilisées (entre parenthèses) ainsi que le nombre de paramètres à régler/ajuster. De plus, toutes les fonctions OpenCV citées sont détaillées en [annexe](#) afin de bien comprendre le fonctionnement de ces dernières.

### 7.1 Détection des centres des plants de betteraves

#### 7.1.1 Présentation de l'algorithme



**① Image RGB :**

Récupération de l'image RGB à partir du topic de la caméra.



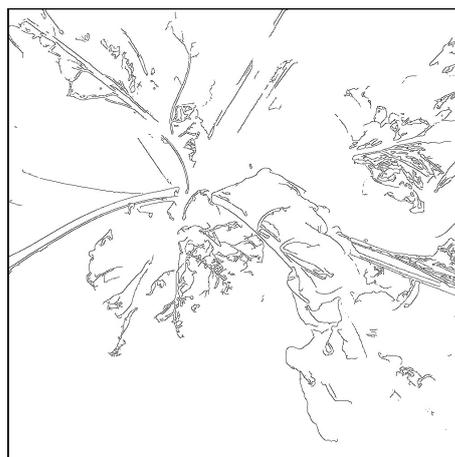
**② Image en niveau de gris :**

Conversion de l'image en niveau de gris à partir de l'image RGB. (`cvtColor`)



**③ Filtre HSV :**

On convertit l'image RGB en image HSV (`cvtColor`) et on la seuille selon H(30-80), S(0-255) et V(0-255) (`Scalar` et `inRange`). (7 paramètres)



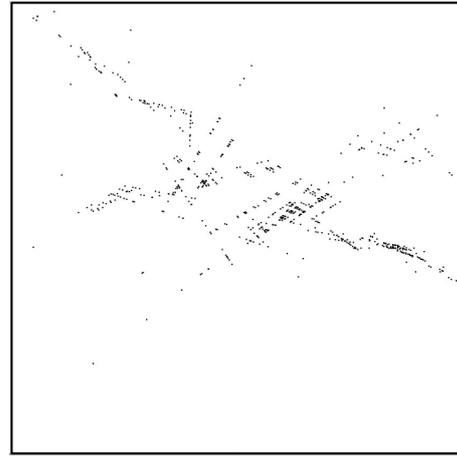
**④ Filtre de Canny :**

On « floute » un peu l'image en niveau de gris (`GaussianBlur`) puis on détecte les contours (`canny`) et on ne garde que les contours verts grâce à l'image HSV (`copyTo`). (3 paramètres)



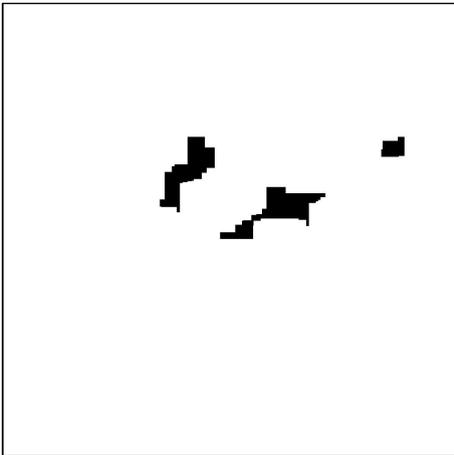
**⑤ Fonction HoughLines :**

On détecte les lignes sur l'image obtenue précédemment, (`HoughlinesP`) puis on les allonge d'une certaine longueur. (4 paramètres)



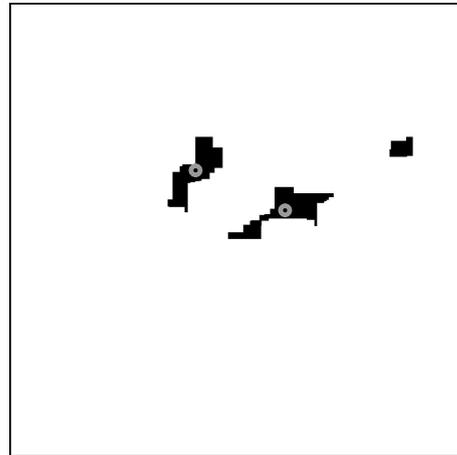
**⑥ Calcul des points d'intersection :**

On calcule les points d'intersection de ces lignes. Ici, l'image obtenue est une image binaire contenant plusieurs pixels noirs, chacun d'eux représentant une intersection.



**⑦ Fonction Dilation et érosion :**

On va dilater (`dilate`) puis éroder (`erode`) plusieurs fois les pixels noirs sur l'image. (3 paramètres)



**⑧ Fonction Choix des surfaces :**

On calcule, pour chaque groupe de pixels, son aire et on détermine la plus grande surface. Ensuite, si un groupe de pixels a sa surface supérieure à un certain pourcentage de la plus grande surface calculée, le groupe de pixels est retenu et on récupère son centre dans l'image (`ConnectedComponents`) (1 paramètre)



**⑨ Fonction Tensorflow :**

On prend une photo d'une taille 150 x 150 autour de chaque centre. On envoie cette image à l'entrée du réseau de neurones. Si, d'après le réseau de neurones, l'image est un centre de bettrave alors le centre est affiché sur l'image RGB et est considéré comme un centre valable. (1 paramètre)

### 7.1.2 Utilisation du réseau de neurones Inception V3

Trois classes constituent le modèle : « Centre de betterave », « Feuille de betterave » et « Fond » (Fond contient des images comportant entre autres de la terre, des cailloux etc...). Les images utilisées lors de l'entraînement du modèle proviennent des expérimentations réalisées et de photos fournies par l'ITB.

Pusieurs images ont été ajoutées pour chaque classe : 84 images pour la classe « Centre de betterave », 99 images pour la classe « Feuille de betterave » et finalement 33 images pour la classe « Fond ».

Le réseau de neurones a donc été entraîné à partir de ces images. Le but étant que lorsque l'on envoie une image en entrée de celui-ci, il doit être capable de déterminer s'il s'agit soit d'un centre de plant de betterave, soit d'une feuille ou d'une image de fond.

Le réseau est utilisé de la manière suivante : nous envoyons une image en entrée et nous obtenons le résultat du modèle en sortie, on obtient un score entre 0 et 1 pour chaque classe associée à l'image envoyée. Par exemple, si pour une image envoyée le résultat est :

« Centre de betterave » = 0.94, « Feuille de betterave » = 0.05, « Fond » = 0.01

Alors, le meilleur résultat est obtenu pour la classe « Centre de betterave » donc le modèle estime que c'est un centre.

Dans le code existant, il suffit que le résultat obtenu pour la classe « Centre de betterave » soit supérieur aux deux autres pour qu'il soit considéré comme valable.

### 7.1.3 Critique et analyse de l'algorithme existant

On rappelle que l'on désire créer un algorithme robuste aux différentes contraintes environnementales puisque le robot travaillera dans un environnement extérieur et sera donc notamment sensible aux variations de luminosité. On va énumérer ici les avantages et les inconvénients de l'algorithme mis en place. L'image présentée en exemple dans la présentation de l'algorithme (7.1.1) est tirée d'une vidéo enregistrée (via un fichier .bag obtenu avec ROS).

L'utilisation du réseau de neurones « Inception V3 » a l'avantage de gagner du temps par rapport à la programmation d'un nouveau réseau de neurones pour la reconnaissance d'objets dans une image.

Pour l'entraînement du réseau de neurones, nous avons moins de 100 images pour chaque classe (« Centre de betterave », « Feuille de betterave » et « Fond »). Or, lorsque le volume des données est faible, cela peut entraîner un sur-apprentissage, c'est-à-dire que le réseau de neurones s'est ajusté parfaitement aux données sur lesquelles il s'est entraîné et ne donne pas de bons résultats lorsqu'une nouvelle image est présentée.

Il faudrait donc constituer une base de données avec plusieurs centaines d'images pour améliorer le modèle et éviter le sur-apprentissage.

Le programme de traitement d'images comporte près de 20 paramètres à régler. Ces paramètres interfèrent les uns sur les autres et il est difficile de trouver un jeu de paramètres fixes permettant d'obtenir de bons résultats. Il est souvent nécessaire de faire varier certains paramètres à cause de la luminosité changeante par exemple.

Néanmoins, Il est possible de trouver un jeu de paramètres qui puisse fonctionner pour une image donnée mais il n'est pas possible de trouver un jeu de paramètres fixes fonctionnant pour différentes images.

De plus, les contours trouvés à partir de la fonction « canny » dépendent du flou appliqué sur l'image et de la luminosité. Dans certains cas, soit on n'obtient pas suffisamment de contours de betteraves, on ne les détecte pas entièrement soit on en détecte trop. Ce qu'il faut retenir c'est que les résultats de cette fonction conditionnent tous les résultats suivants.

Enfin, le travail de bibliographie existant effectué jusque là n'a pas été, de mon point de vue, assez poussé. Le sujet mériterait des recherches complémentaires.

Nous avons donc deux solutions à notre disposition :

- Si l'on veut garder l'algorithme existant, il faut trouver une méthode qui permette un réglage automatique des paramètres en fonction de l'image d'entrée et de la luminosité.
- Sinon, il faut mettre en place de nouvelles méthodes de détection et de segmentation afin d'obtenir un comparatif permettant de dégager la meilleure solution.

## 7.2 Segmentation des feuilles de betteraves

L'algorithme existant présenté ici est en grande partie inspiré du tutoriel proposé par « OpenCV » pour faire de la segmentation en utilisant les algorithmes « DistanceTransform » et « Watershed » que l'on peut retrouver à l'adresse suivante : [https://docs.opencv.org/3.4/d2/dbd/tutorial\\_distance\\_transform.html](https://docs.opencv.org/3.4/d2/dbd/tutorial_distance_transform.html)

### 7.2.1 Présentation de l'algorithme



① **Image RGB :**  
Récupération de l'image RGB à partir du topic de la caméra.



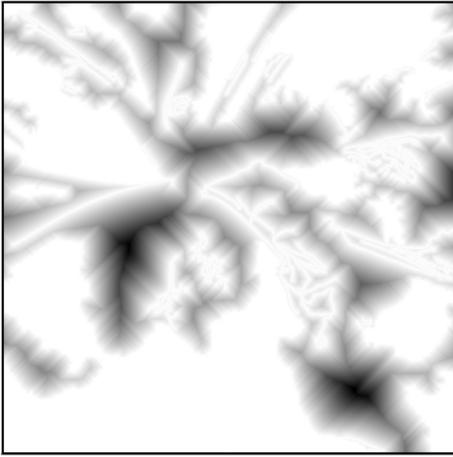
② **Image en niveau de gris :**  
Conversion de l'image en niveau de gris à partir de l'image RGB. (`cvtColor`)



③ **Filtre RGB :**  
On seuille l'image RGB en fonction des proportions entre les taux R, G et B, ( $G > \frac{1}{3}(R + G + B)$ ) puis on filtre le bruit. (`ConnectedComponents`)



④ **Filtre de Canny :**  
On « floute » l'image en niveau de gris (`GaussianBlur`) puis on détecte les contours (`canny`). On utilise aussi l'image RGB seuillée, sur laquelle on ajoute les contours trouvés. (`addWeighted`)



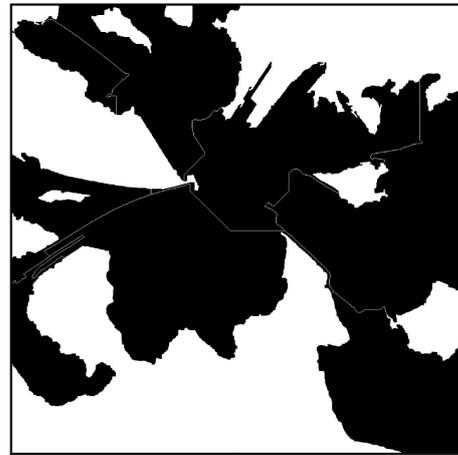
⑤ **Fonction Carte de distance (1) :**  
 A partir de l'image précédente, on calcule, pour chaque pixel noir dans l'image, la plus petite distance de ce pixel à un pixel blanc. La distance sera représentée par une valeur en niveau de gris proportionnelle à la distance. Si un pixel noir est très loin de son plus proche pixel blanc, sur l'image de sortie, ce pixel aura une valeur en niveau de gris élevée (proche du noir) et inversement. ([distanceTransform/normalize](#))



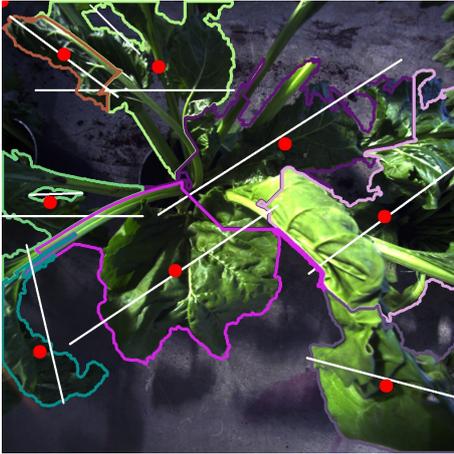
⑤ **Fonction Carte de distance (2) :**  
 On seuille l'image pour obtenir une image binaire. Le seuil correspond à la valeur de niveau de gris pour lequel on accepte qu'un pixel soit blanc ou noir. En jouant sur la valeur de ce seuil, on sépare plus ou moins les feuilles ([thresh](#)). Enfin, on ajoute tous les pixels blancs trouvés sur l'image RGB seuillée ③. ([copyTo](#))



⑥ **Fonction récursive :**  
 La fonction récursive va, pour chaque groupe de pixels noirs, déterminer si son aire est supérieure à 10 000 pixels<sup>2</sup>. Si c'est le cas, on sépare ce groupe de pixels en plusieurs petits groupes, puis on rappelle la fonction à nouveau pour vérifier la condition sur l'aire et ainsi de suite jusqu'à ce que tous les groupes de pixels aient leur aire inférieure à la limite fixée.



⑦ **Fonction Algorithme de Watershed :**  
 Tout d'abord, on récupère sur l'image précédente, les contours de chaque groupe de pixels noirs sous forme de vecteurs de points ([findContours](#)) puis on les dessine sur une nouvelle image. ([drawContours](#)) Ensuite, à partir de cette image et de l'image issue de ④, on étend chacun de ces contours jusqu'à ce que l'un d'eux se rencontre ou qu'il rencontre une ligne de séparation (blanche) issue de ④. ([watershed](#))



### ⑧ Fonction Calcul des centres :

Pour chaque zone, on assigne une couleur aléatoire puis on calcule le centre (`ConnectedComponents`) ainsi que l'axe d'inertie. Chaque point rouge correspond à un centre de feuille trouvé; les axes d'inertie de chaque feuille sont également ajoutés à l'image RGB initiale.

## 7.2.2 Critique et analyse de l'algorithme existant

Dans cette analyse, on remarquera une similitude au niveau des problèmes rencontrés dans la détection des centres via les fonctions utilisées.

Tout d'abord, on peut citer l'utilisation de la fonction « canny », qui, comme dit précédemment, est dépendante de la luminosité et dont les résultats conditionnent le reste du programme.

On ignore pourquoi il y a une différence entre le seuillage HSV pour la détection des centres et un seuillage RGB pour la détection des feuilles alors que l'on veut le même résultat à la sortie de cette fonction, c'est-à-dire segmenter toute la plante à partir de sa couleur (verte).

Contrairement à l'algorithme de détection des centres, les paramètres sont moins nombreux. Néanmoins le réglage de ceux-ci est à faire au cas par cas pour obtenir un bon résultat (un point par feuille).

Par exemple, dans la fonction récursive ⑥ l'aire maximale d'une feuille va dépendre du nombre de pixels noirs sur celle-ci, donc de la distance à laquelle se trouve la caméra pour regarder les feuilles. Si la caméra est trop proche des feuilles, les aires trouvées seront très grandes et trop loin, les aires trouvées seront trop petites.

On peut aussi remarquer que l'algorithme segmente toute la plante et pas uniquement les feuilles. En général, le centre de la betterave est considéré comme une partie d'une feuille. Il faudrait une segmentation plus précise.

De la même manière que la détection des centres de plants de betteraves, il n'y a pas eu d'analyses comparatives entre les différentes méthodes pour faire de la segmentation afin de trouver le meilleur algorithme répondant aux contraintes fixées par ce projet.

Nous avons donc deux solutions à notre disposition :

- Si l'on veut garder l'algorithme existant, il faut trouver une méthode qui permette un réglage automatique des paramètres en fonction de l'image d'entrée, de la luminosité et de la taille/forme des feuilles.
- Sinon, il faut trouver de nouvelles méthodes pour la détection et la segmentation des feuilles afin d'avoir un comparatif permettant de dégager la meilleure solution.

### 7.3 Contrôle/Commande du robot

Les travaux présentés ci-après sont les travaux réalisés en simulation via Gazebo.

Le but est de commander le bras robotisé au-dessus de chaque feuille. Dans la simulation, la feuille à atteindre est représentée par un cylindre vert. Le travail effectué jusque là permet de détecter le cylindre dans l'image puis de déplacer le robot au-dessus de ce dernier. Nous allons uniquement détailler la partie existante que je vais modifier pour réaliser le travail à effectuer (Estimation de la pose désirée). La commande du bras et de l'axe linéaire étant déjà réalisée, elle ne sera pas modifiée.

#### 7.3.1 Fonctionnement général de l'algorithme de commande

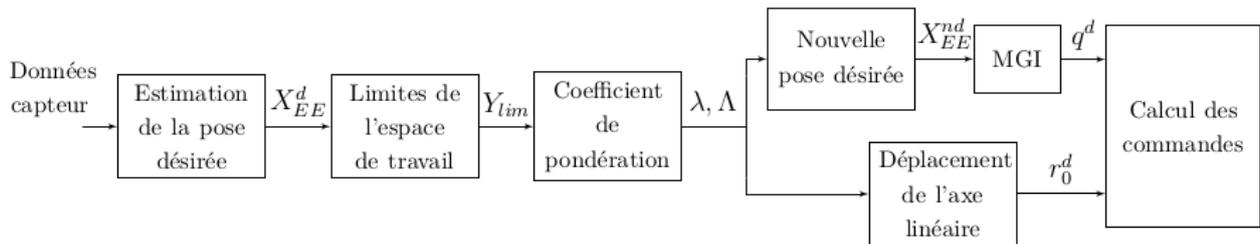


FIGURE 3 – Fonctionnement de l'algorithme

La figure ci-dessus schématise le fonctionnement des algorithmes implémentés. Tout d'abord, on estime la future pose à atteindre pour se centrer au-dessus d'une feuille à partir des informations capteurs (traitement d'image). Une fois cette pose déterminée, il est possible de calculer les limites de l'espace de travail selon l'axe de redondance, ici  $Y_{lim}$ . Ensuite, le coefficient de pondération  $\lambda$  est calculé, puis une matrice de pondération  $\Lambda$  est établie, permettant de dissocier l'impact des redondances sur les différents axes. Enfin, il est possible de calculer les déplacements à effectuer sur chacune des deux parties du système (bras manipulateur et axe linéaire), et de générer les commandes correspondantes.

#### 7.3.2 Description de l'algorithme de détection du cylindre

Pour détecter le cylindre, l'image est seuillée via un seuillage HSV pour récupérer les pixels correspondant au cylindre. On obtient donc une image binaire dont les pixels noirs correspondent au cylindre et les pixels blancs au reste (fond). A partir de cette image, on peut obtenir le cercle circonscrit au cylindre.

On récupère ensuite le rayon et le centre du cercle ( $XG, YG$ ) dans l'image.  $R$  correspond à la distance (en pixels) entre le centre du cylindre et le centre de l'image.

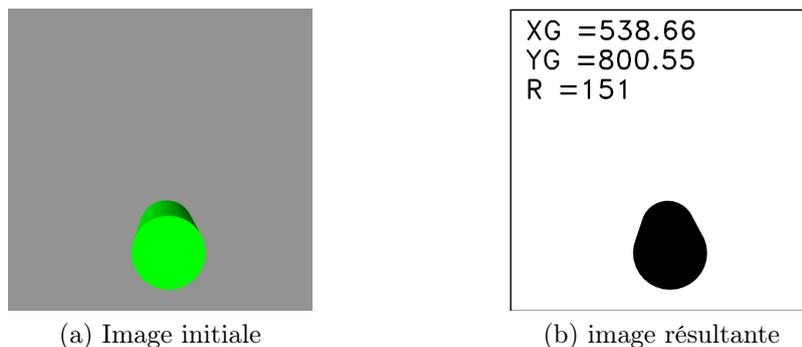


FIGURE 4 – Résultats de l'algorithme de détection du cylindre

### 7.3.3 Description de l'algorithme de détection lié à la commande

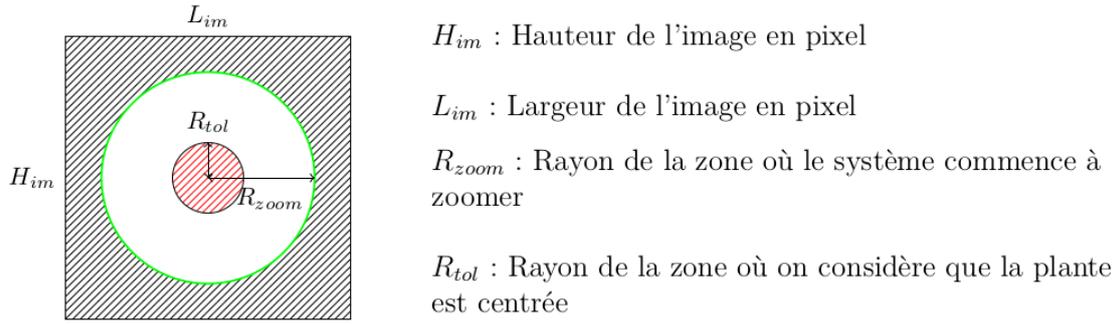


FIGURE 5 – Décomposition de l'image

A partir de la position  $(XG, YG)$  correspondant au centre du cylindre dans l'image, une consigne proportionnelle à la distance en pixel entre le centre de l'image et le centre du cylindre détectée est définie, telle que :

$$\begin{cases} d_x = -\left(YG - \frac{H_{im}}{2}\right) \alpha \\ d_y = \left(XG - \frac{L_{im}}{2}\right) \alpha \end{cases}$$

avec  $d_x$  et  $d_y$  le déplacement selon l'axe x et y respectivement,  $H_{im}$  et  $L_{im}$  la hauteur et la largeur de l'image en pixel et  $\alpha$  un coefficient de proportionnalité égal à  $1/3000$ . L'échange entre les différents axes x et y est dû au fait que le repère caméra n'est pas aligné avec le repère de l'effecteur.

Une zone de tolérance d'un rayon modifiable a été définie afin d'éviter que le bras oscille au-dessus de la feuille  $R_{tol}$ . Sur la figure 5, cette zone est représentée en rouge. Puisque d'une itération à l'autre, le centre de la feuille n'est pas spécialement détecté sur le même pixel.

On peut aussi venir estimer un déplacement selon l'axe z qui correspond à la hauteur de la caméra. Pour ce faire, une deuxième zone de l'image a été définie. En s'appuyant sur la figure 5, on peut définir le comportement du système comme le suivant : si une feuille est détectée dans la zone hachurée en noir, des déplacements sont générés uniquement selon les axes x et y.

En effet, si la caméra commence à se rapprocher d'une feuille alors qu'elle est située près d'un bord de l'image, il est possible de la faire sortir du cadre de l'image. Si le centre de la feuille est situé à l'intérieur du cercle vert et à l'extérieur du cercle rouge, on vient alors générer des mouvements selon les trois axes, afin de centrer la caméra et de la rapprocher au-dessus de la feuille. Enfin, si le centre de la feuille est situé dans le cercle rouge, on considère que la feuille est suffisamment centrée, et donc on vient juste générer une commande de zoom selon l'axe z si besoin.

Le déplacement selon l'axe z est alors calculé de la manière suivante :

$$\begin{cases} d_z = \left(R_{cercle}^d - R_{cercle}\right) \alpha & \text{si } R < R_{zoom} \\ d_z = 0 & \text{sinon} \end{cases}$$

en notant  $R_{cercle}$  le rayon du cercle inscrit mesuré,  $R_{cercle}^d$  le rayon du cercle inscrit désiré et R la distance entre le centre de l'image et le centre du cylindre.

A partir de  $d_x$ ,  $d_y$  et  $d_z$ , la nouvelle position désirée est calculée.

## 8 Amélioration des programmes existants

### 8.1 Détection des centres des plants de betteraves

Le code existant a été réécrit et entièrement commenté pour une meilleure compréhension. Plus de 200 images ont été ajoutées pour chacune des classes « Centre de Betterave », « Feuille de Betterave » et « Fond ». Cette augmentation de données n'a pas vraiment eu d'incidence sur l'efficacité de l'algorithme puisque les problèmes majeurs proviennent du fait qu'il est difficile de régler les paramètres du programme. Après différents tests, nous n'avons pas trouvé de moyens permettant un réglage automatique des paramètres en fonctions des différentes conditions dans lesquelles les plantes évoluent.

### 8.2 Segmentation des feuilles de betteraves

Comme pour l'algorithme de détection des centres, le code existant a été réécrit et entièrement commenté. Aucune corrélation n'a été trouvée entre les valeurs des paramètres, les conditions lumineuses variables et les tailles/formes des feuilles.

En conséquence, ne disposant pas de solutions pour automatiser les paramètres du programme actuel en fonction de la luminosité, il a été décidé de rechercher un nouvel algorithme pour la détection de feuilles.

## 9 Recherches d'un nouveau algorithme pour la segmentation de feuilles

### 9.1 Recherches bibliographiques

Dans cette recherche bibliographique, nous avons voulu tout d'abord avoir une vision d'ensemble sur les méthodes de segmentation d'objets, puis plus particulièrement pour la segmentation de feuilles afin de déterminer les méthodes les plus efficaces et robustes.

Tout d'abord, nous nous attacherons à présenter les différentes études traitant des méthodes existantes de segmentation d'objets dans une image. Ensuite, je détaillerai les études les plus récentes pour la segmentation des feuilles en particulier. Enfin, nous concluerons cette analyse bibliographique en déterminant les meilleures méthodes à mettre en oeuvre pour ce projet.

Le papier [18] est une synthèse des différentes méthodes de deep learning pour la classification d'images, la détection d'objets en général ainsi que les différentes bases de données existantes. Il est décrit l'architecture de différents réseaux et des bases de données de différents types (2D, 2.5D, 3D). L'une d'entre elles est la base de données COCO [32] (Microsoft Common Objects in Context) qui comprend plus de 80 classes d'objets pour un total d'environ 80 000 images. C'est une base comportant des objets divers et communs (de la vie de tous les jours).

D'après leur analyse, les auteurs concluent que la plupart des résultats sont difficilement reproductibles, par manque de code source disponible et le manque d'informations concernant le temps d'exécution ou la mémoire allouée. Toujours d'après leurs conclusions, Deeplab [10] est la meilleure méthode pour faire de la segmentation sur des images RGB.

Ce papier [22] fait état de différentes techniques de segmentation possibles utilisées pour l'analyse et les classe selon 3 catégories en se basant sur des méthodes de segmentation sur l'indice de couleur, sur le seuillage (basé sur la morphologie de la plante) et enfin des méthodes basées sur l'apprentissage.

Il en résulte que les méthodes de segmentation basées sur la couleur sont simples à implémenter, ont un temps de calcul rapide mais ne sont pas robustes à des conditions peu ou très ensoleillées. Les méthodes basées sur le seuillage à partir de la morphologie de la plante sont plutôt efficaces, ont un temps de calcul correct mais nécessitent des réglages en fonction de la luminosité. Enfin les méthodes basées sur l'apprentissage ont montré des résultats plus fiables qu'avec les méthodes précédentes mais nécessitent une phase d'apprentissage et ont un temps de calcul plus long pour le traitement des images.

Le document [43] compare 4 méthodes de segmentation utilisées lors du CVPPP 2014 avec les différentes métriques associées à celui-ci. Tout d'abord, avec une segmentation en appliquant des histogrammes 3D [34] puis en utilisant la segmentation par superpixels [2], par Chamfer matching [6] et enfin avec l'algorithme de Watershed [48].

Pour ce qui est de segmenter/discriminer la plante du fond, la segmentation par superpixels et via les histogrammes 3D montrent une meilleure robustesse. Pour ce qui est de segmenter/compter les feuilles, chaque algorithme présente des résultats aberrants soit par sous-segmentation ou sursegmentation. Par exemple, l'algorithme de Watershed a tendance à sursegmenter et à considérer comme étant des feuilles, d'autres parties de l'image qui ne le sont pas.

### Le séminaire CVPPP 2018 et les challenges LSC et LCC

CVPPP<sup>3</sup> (Computer Vision Problems in Plant Phenotyping) est un séminaire organisant les challenges LSC(Leaf Segmentation Challenge) et LCC(Leaf Counting Challenge) en partenariat avec le ECCV (European Conference on Computer Vision).

Le but des challenges LSC et LCC est de continuer à étendre l'état de l'art de la vision par ordinateur pour le phénotypage de plantes.

Les résultats présentés ici proviennent entre autre de l'éditorial CVPP 2018 qui recueille les derniers travaux de recherches pour l'année 2018 [36].

[42], [33], [7] correspondent aux bases de données utilisées pour le LSC (Leaf Segmentation Challenge) pour le CVPPP 2014 avec des plants de tabac et de Arabidopsis (Arabette des dames). Il y a au final 3 bases de données (A1 : 164 images (Arabette des dames) ; A2 : 201 images (Arabette des dames) ; A3 : 83 images (Tabac)) Les images ont été prises dans des conditions lumineuses bien précises qui y sont détaillées.



(a) image issue de A1



(b) image issue de A2



(c) image issue de A3

FIGURE 6 – Exemples d'images des bases de données A1, A2, A3 (images tirées de [42])

3. <https://www.plant-phenotyping.org/CVPPP2017-challenge>

Les auteurs de [49] ont créés 2 bases de données d'images synthétiques : A4 et A5. En utilisant les datasets du LSC et en ajoutant des données synthétiques à partir des données initiales, ils obtiennent les meilleurs résultats sur les A1 et A2. Leurs résultats sur le réseau Mask-RCNN montrent des exemples d'invariance à la luminosité. Cette méthode a l'avantage de pouvoir être rapidement appliquée sur une autre plante.

Les auteurs de [52] utilisent un réseau de neurones GAN pour générer des images synthétiques qu'ils ajoutent pour l'entraînement du réseau Mask R-CNN. L'extension des données a permis de diminuer l'erreur moyenne de comptage de feuilles.

Les auteurs de [27] ont comparé 3 méthodes pour faire de la segmentation de feuilles de soja : le Random Forest Classifier [9], un seuillage couleur via HSV en utilisant la méthode de Otsu [47] et enfin le réseau de neurones DeeplabV3 [10]. La base de données utilisée contient 285 images de soja en champs dans différentes conditions de luminosité.

Les images pour l'entraînement ont été annotées avec l'outil LabelMe Annotation Tool [40].

Ils ont aussi augmenté le nombre d'images en effectuant entre autre des rotations de 90°.

Les résultats obtenus par deeplab V3 donnaient des résultats plus cohérents. Ils précisent que le mieux serait d'augmenter le nombre de données d'apprentissage pour améliorer le réseau.

Par ailleurs, d'autres études ayant modifié certains réseaux de neurones existants montrent des résultats intéressants, comme par exemple avec le réseau Resnet 38 [13], Resnet 50 [14], Resnet 101 [41] ou même avec des réseaux convolués spécialement créés pour répondre au problème [3]. Toutefois, comme dans plusieurs études, il n'est pas possible de mettre en oeuvre les méthodes mises en place car les programmes utilisés ne sont pas en libre accès. Il est donc difficile d'en vérifier les résultats ou de les implémenter sur d'autres données sans avoir à refaire les modifications effectuées.

Il faut savoir qu'il existe bien d'autres outils que les caméras RGB pour faire du phénotypage de plante [30]. De plus, il n'existe pas de base de données contenant des feuilles de betteraves hormis dans celle-ci [31] mais, se limitant à un certain stade de croissance, elle ne permet pas la détection de maladies. D'autres études sont effectuées dans des conditions d'expérience très particulières [51] autre que dans un champ.

Globalement les meilleurs résultats sont obtenus via l'utilisation de réseaux de neurones au détriment du temps de calcul.

Certaines études ont montré que l'augmentation des données d'entraînement donne toujours de meilleurs résultats, une meilleure robustesse du réseau. Au vu des résultats récemment obtenus lors du concours LSC, deux méthodes ont retenu notre attention pour faire de la segmentation : le réseau Deeplab [10] et Mask-RCNN [23].

Le réseau Mask R-CNN a été choisi car les résultats donnés en sortie du réseau contiennent à la fois l'encadrement et le contour de l'objet tandis que Deeplab ne donne que le contour.

## 9.2 Implémentation du réseau Mask R-CNN

D'après les recherches effectuées, nous en avons déduit que le réseau Mask R-CNN faisait partie des meilleures solutions trouvées pour faire de la segmentation de feuilles. Nous considérons que si le réseau s'entraîne sur des cas nombreux et variés (petites et grandes feuilles avec luminosité variable) l'algorithme sera robuste à ces différentes variations.

Nous avons utilisé l'implémentation de Mask R-CNN sous Keras et Tensorflow [1] en utilisant notre propre base de données contenant des images de plants de betteraves.

L'architecture du dossier Mask R-CNN sera détaillée en [annexe](#).

### 9.2.1 Présentation du réseau

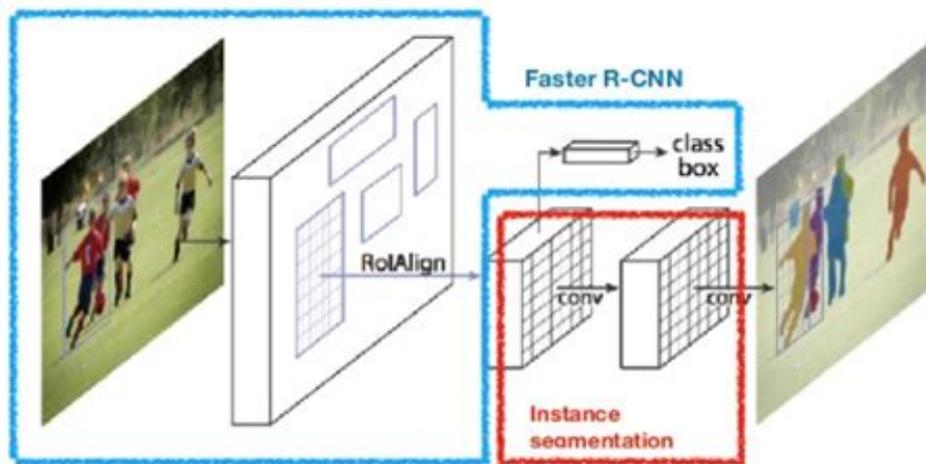


FIGURE 7 – Architecture du réseau Mask-RCNN(image tirée de [23])

Le réseau Mask R-CNN a subi plusieurs améliorations au cours des dernières années.

#### **R-CNN<sup>4</sup> [20](2014) :**

Le réseau R-CNN a été créé par une équipe de l'UC Berkeley composée de Jitendra Malik, Ross Girshick, Jeff Donahue and Trevor Darrel.

Ce réseau détecte plusieurs régions d'intérêt dans l'image puis essaie d'associer chaque région à une classe parmi celle de la base de données PASCAL VOC [16].

#### **Fast R-CNN [19] (2015) :**

En 2015, Ross Girshick a modifié le réseau pour accélérer la classification des régions proposées.

#### **Faster R-CNN [38](2016) :**

En 2016, une équipe de de recherche Microsoft composée de Shaoqing Ren, Kaiming He et Ross Girshick ont modifié la méthode utilisée pour la détection de régions d'intérêt.

#### **Mask R-CNN [23](2017) :**

Enfin, l'ajout d'un réseau complètement connecté a permis de faire de la segmentation pixel par pixel et a nécessité la modification de la classification d'objets (RoIAlign) pour être alignée avec la segmentation par pixel.

---

4. R-CNN : Regional Convolved Neural Network

### 9.2.2 Entraînement du réseau

Pour réaliser l'entraînement du réseau, nous avons à notre disposition, 3 vidéos comprenant des plants de betteraves en champs obtenues par un autre robot que celui du projet Phenafol. Nous avons donc constitué 3 groupes d'images comme suit :

67 images à partir de la vidéo n°1, 86 images à partir de la vidéo n°2 et 29 images à partir de la vidéo n°3.

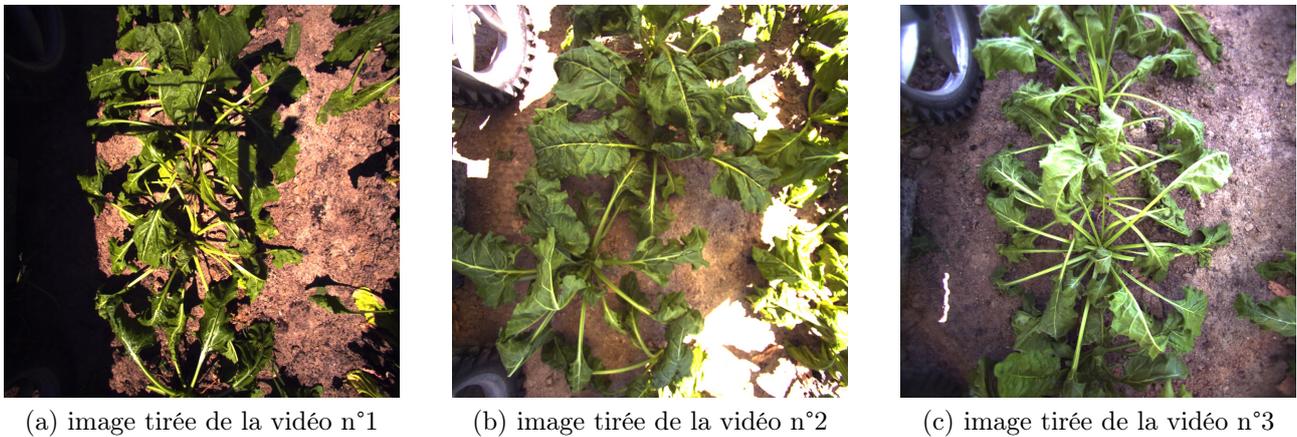


FIGURE 8 – Exemples d'images tirées des différentes vidéos

Ensuite, parmi ces 182 images, nous en avons sélectionné 30 de façon aléatoire et constitué 2 catégories : 20 images pour l'entraînement du réseau et 10 images pour la validation. Le test du réseau a été réalisé sur ces 10 images.

Pour que le réseau s'entraîne, il faut segmenter manuellement toutes les feuilles dans chacune des 20 images. Pour cela, j'ai utilisé l'outil VGG Annotator [15]. Grâce à cet outil, on peut encadrer chaque feuille sous forme de polygones qui correspondront donc aux contours de chacune d'elles. Dans l'image ci-dessous, les contours sont de couleur jaune.



FIGURE 9 – Exemple d'une image segmentée avec VGG Annotator et utilisée pour l'entraînement

Puisque chaque image contient environ une vingtaine de feuilles, nous avons donc approximativement 400 feuilles sur lesquelles s'est entraîné le réseau.

Une fois que les 20 images sont annotées, on peut enregistrer ces annotations sous forme d'un fichier .json qui comprendra donc les coordonnées de chaque contour pour chacune des 20 images. Le fichier .json doit se trouver dans le dossier train<sup>5</sup> contenant également les 20 images pour l'entraînement. C'est à partir de ce fichier que le réseau peut être entraîné.

On répète la même opération pour les 10 images de validation et on ajoute le fichier .json associé dans le dossier eval. Une fois les données préparées, nous pouvons entraîner le réseau.

Plusieurs paramètres peuvent être réglés pour l'entraînement :

- les poids initiaux du réseau au début de l'entraînement,
- le nombre de rétropropagation,
- le nombre de pas pour chaque rétropropagation,
- le nombre d'images par GPU<sup>6</sup>.

Il est conseillé d'utiliser les poids de la base de données COCO pour l'initialisation plutôt que d'initialiser les poids aléatoirement. Nous avons donc choisi les poids de la base COCO.

Plus le nombre de rétropropagations ainsi que le nombre de pas associé est grand, plus le réseau donnera de meilleurs résultats (au moins sur la base sur laquelle il s'est entraîné) et plus le temps d'entraînement sera long. Nous avons gardé les paramètres qui étaient par défaut soit 20 rétropropagations de 100 pas chacun.

Le nombre d'images par GPU correspond au nombre d'images que le GPU va pouvoir traiter simultanément. Plus ce nombre est élevé, plus le temps d'entraînement est rapide. On choisit ce nombre en fonction de la capacité de mémoire du GPU.

Le GPU utilisé avait une capacité de mémoire de 4Gb.

Celui-ci n'ayant pas une grande capacité de mémoire, nous avons mis 1 image par GPU. 2 images par GPU provoquaient une erreur et l'entraînement ne s'effectuait pas.

Dans notre cas, l'entraînement du réseau a pris une douzaine d'heures.

Lorsque le réseau a terminé de s'entraîner, nous pouvons récupérer les nouveaux poids du réseau entraîné qui servent à faire les tests.

### 9.2.3 Tests et résultats

Le temps d'exécution pour une image est compris entre 4 et 6 secondes. Cela est dû au fait que l'image à traiter est grande (1022x1022) et que la mémoire du GPU est faible (4Gb). L'implémentation du réseau est fait via Keras et Tensorflow avec le langage Python.

Chaque image résultante présente 3 caractéristiques :

- l'encadrement de toutes les zones détectées comme étant des feuilles,
- le contour de chacune des feuilles,
- le score de prédiction associé,
- la classe associée (ici il n'y a qu'une seule classe, la classe « feuilles »).

---

5. Pour plus d'informations, l'architecture des dossiers du réseau Mask R-CNN est détaillée en [annexe](#).

6. GPU : General Processor Unit

Pour chacun des tests, il est possible de régler le paramètre « score de prédiction ». Par exemple, si ce paramètre est initialisé à 0.9, le réseau affichera sur l'image de sortie uniquement les feuilles trouvées dont le score est supérieur ou égal à 0.9.

Le réseau ne donne pas de résultats inférieurs à 0.5 si ce paramètre est inférieur à 0.5.

De plus, la modification de ce paramètre n'a pas d'influence sur le temps de traitement de l'image.

Nous avons la possibilité de pouvoir comparer les résultats prédits par le réseau à partir des images annotées dans le dossier de validation.

Le réseau donne une « précision moyenne » des résultats obtenus entre 0 et 1. Un exemple détaillé du calcul de la précision moyenne est donné en [annexe](#). Sur les 10 images de validation, le score est d'environ 0.58. Les images issues de la vidéo 1 (comprenant beaucoup d'ombres) ont un score d'environ 0.4 tandis que les images issues des vidéos 2 et 3 (ensoleillement correct) ont un score d'environ 0.7.

Les images contenant beaucoup d'ombres obtiennent en général de moins bons résultats. Cela est dû au fait que le nombre de données pour l'entraînement est faible et que le réseau n'est pas encore robuste à ces différentes variations de luminosité.



(a) Image initiale en entrée du réseau



(b) Image résultante en sortie du réseau

FIGURE 10 – Exemple de résultat du réseau Mask R-CNN

La base de données d'entraînement sera enrichie par de nouvelles images issues du robot Phenafol qui permettront d'une part, d'augmenter la robustesse du réseau mais aussi de coller au plus près des conditions dans lesquelles le robot va travailler. De plus, des recherches complémentaires seront effectuées pour trouver une implémentation du réseau Mask-RCNN sous C++ afin d'améliorer la rapidité de détection.

# 10 Recherches d'un algorithme pour le suivi d'objets dans une image

## 10.1 Recherches bibliographiques

Pour faire du tracking sur une feuille, il faut rechercher une méthode qui réponde dans la mesure du possible aux exigences suivantes :

- un temps de calcul rapide (au moins 20 FPS<sup>7</sup>),
- une insensibilité aux variations de luminosité,
- une robustesse aux changements de forme ou de taille de la feuille au fur et à mesure que la caméra se rapproche de celle-ci.

Dans cette recherche bibliographique, j'ai voulu avoir une vision d'ensemble sur les méthodes de tracking d'objets afin de déterminer la solution à mettre en oeuvre. Différentes études faisant une revue des méthodes existantes pour le tracking d'objet ont été analysées.

Dans plusieurs études, [35], [45], [44], trois grands thèmes sont abordés : la détection d'objets, la classification d'objets et le suivi d'objets. Dans notre cas, la détection et la classification sont effectuées par le réseau Mask R-CNN. Nous allons donc nous concentrer uniquement sur les méthodes de tracking qui y sont présentées.

On trouve trois catégories de tracking : le « Point Tracking » (Le suivi d'objets en utilisant un ou plusieurs points précis pour représenter l'objet), le « Kernel-Based Tracking » (Le suivi d'objets en utilisant une zone encadrant l'objet) et le « Silhouette Tracking » (le suivi d'objets en prenant en compte la forme spécifique de l'objet).

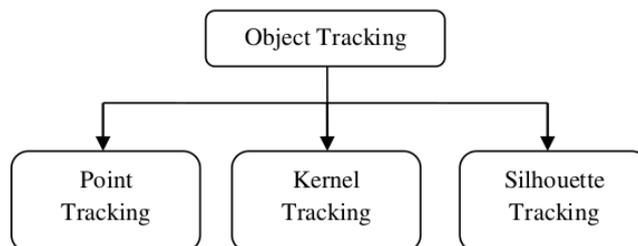


FIGURE 11 – Les différentes catégories de suivi d'objets (image tirée de [35])

Pour chacune de ses catégories, il existe aussi plusieurs méthodes et plusieurs algorithmes différents.

Cette étude [46] présente différentes méthodes existantes pour faire du suivi d'objets et de leurs aires d'applications.

En résumé, les méthodes basées sur le « Point Tracking » sont utilisées pour détecter des objets représentés par des points. Ces méthodes ont l'avantage de ne pas être sensibles aux variations de luminosité mais nécessitent la mise en place d'un algorithme de détection pour chaque image/frame de la vidéo.

Les méthodes basées sur le « Kernel Tracking » sont utilisées pour détecter des objets rigides pouvant être représentés par des rectangles ou des ellipses. La détection de l'objet à suivre est uniquement nécessaire pour la première image. Cependant, cette représentation simple d'un

---

7. FPS :Frame Per Second

objet par un rectangle peut avoir le désavantage de prendre en compte des zones de l'image ne correspondant pas à l'objet à suivre.

Enfin, les méthodes basées sur le « Silhouette Tracking » sont utilisées pour détecter des objets ne pouvant pas être représentés par des points ou des formes géométriques simples. Ces méthodes ont l'avantage d'être plus robustes que les deux méthodes précédentes. Toutefois, ces méthodes ont en général un temps de calcul important.

Dans notre cas, nous allons utiliser une solution parmi la catégorie « Kernel-Based » ou « Silhouette-based ». Il existe une multitude de solutions possibles et il est difficile d'en choisir une parmi toutes celles qui existent. D'autres recherches seront effectuées pour compléter cette bibliographie.

Différentes méthodes devront être testées et comparées à l'aide de vidéos prises en champs par le robot.

En fonction de la solution choisie, nous pourrons adapter les résultats donnés par le réseau Mask R-CNN. Si nous choisissons une solution de la catégorie « Kernel-Based », nous prendrons l'encadrement des feuilles donné par le réseau. Si nous choisissons une solution de la catégorie « Silhouette-based », nous prendrons les contours de chaque feuille.

## 10.2 Tests de différentes solutions pour le tracking

Comme dit précédemment dans la recherche bibliographique pour le tracking, plusieurs méthodes devront être testées afin de terminer la solution finale à utiliser. Cela permettra de mettre en évidence les limites des méthodes testées.

Dans un premier temps, les tests réalisés se font en simulation via le simulateur Gazebo.

Comme pour l'algorithme existant, les cylindres verts dans la simulation vont représenter les feuilles faute d'avoir un modèle représentant une plante.

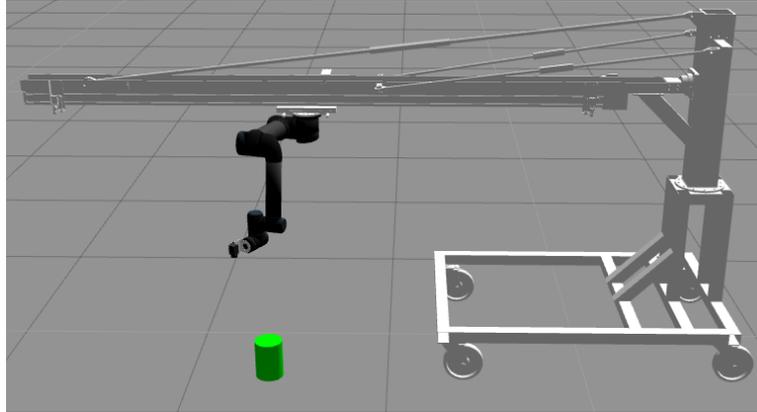


FIGURE 12 – Image du robot dans l'environnement de simulation Gazebo

On rappelle que le travail existant pour la commande du robot permet de déplacer le robot au-dessus d'un cylindre. La détection et la position du centre du cylindre est déterminée sur chaque image en appliquant un seuillage HSV. Ici, le but est de remplacer le seuillage HSV par l'algorithme de tracking pour déterminer la position du cylindre dans chaque image.

La première méthode testée fait partie de la catégorie « Kernel Tracking ». Pour cela nous testons différents algorithmes proposés par OpenCV que l'on peut retrouver par le lien suivant : <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/>

Voici une liste non exhaustive des algorithmes pouvant être testés : « BOOSTING » [21] , « MIL » [4] , « KCF » [24] , « TLD » [26] , « MEDIANFLOW » [25] .

Ces algorithmes sont en cours de tests en simulation. Ils seront aussi testés sur des vidéos prises par le robot en champs en conditions réelles.

### **Première réflexion pour la mise en place de la machine à états du tracking**

Le but de cette machine à états est de simuler le fonctionnement du robot lors de son déplacement au-dessus de chaque feuille. Cette partie étant en cours de développement, cette machine à états est donc susceptible d'être modifiée.

Voici les différents états possibles :

#### **Etat n°1 : Initialisation**

- Déplacement du robot à la position initiale.
- Récupération des informations concernant les feuilles détectées.
- Initialisation de l'algorithme choisi pour le tracking.
- Initialisation de la première feuille à suivre à partir des informations récupérées.

### Etat n°2 : Tracking et déplacement du robot

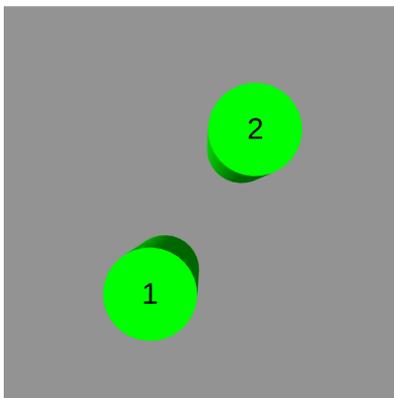
- On récupère les coordonnées de la feuille à détecter dans l'image.
- Mise à jour de l'emplacement de la feuille dans l'image.
- Déplacement du robot vers la nouvelle position désirée.
- Si la caméra est au-dessus de la feuille (la feuille au centre de l'image), on prend une photo et on l'enregistre et on passe à l'état numéro 3.

### Etat n°3 : Déplacement du robot en position initiale

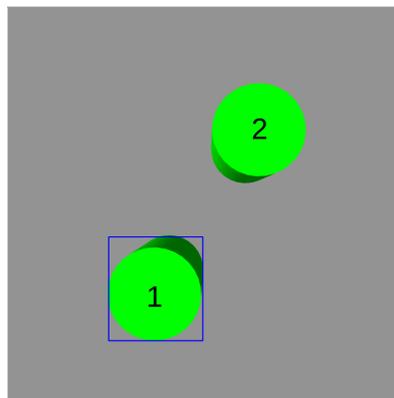
- Déplacement du robot à la position initiale.
- Si le robot est à la position initiale, on passe à l'état 4.

### Etat n°4 : Passage à la feuille suivante ou retour à l'état initial

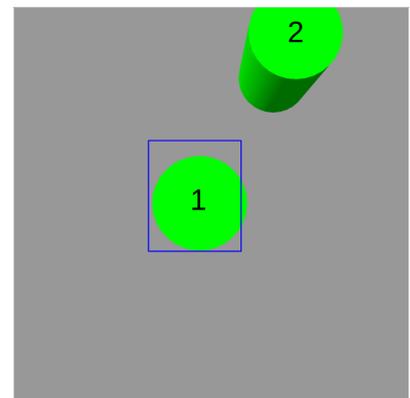
- S'il reste encore des feuilles à traiter, on passe à la feuille suivante et on passe à l'état n°2.
- Dans le cas contraire, on passe à l'état numéro 1.



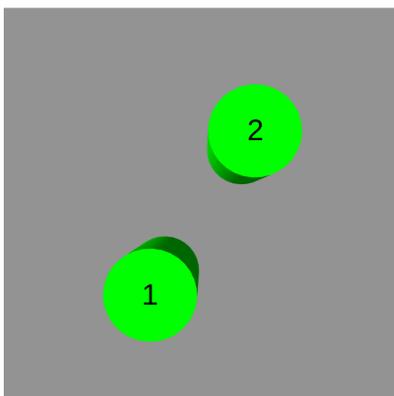
(a) Robot en position initiale



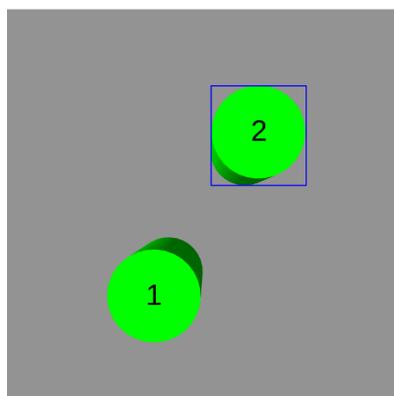
(b) Détection du premier cylindre à suivre



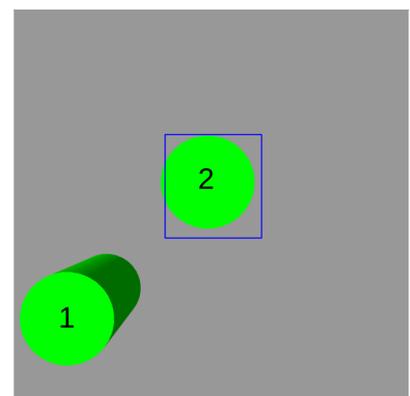
(c) Tracking et déplacement du robot vers le premier cylindre



(d) Retour à la position initiale



(e) Détection du deuxième cylindre à suivre



(f) Tracking et déplacement du robot vers le deuxième cylindre

FIGURE 13 – Visualisation des différents états possibles

# 11 Conclusion

Dans le cadre de ce stage, j'ai pu prendre toute la mesure de la diversité des champs d'application de la robotique, de la perception active, notamment dans un environnement tout nouveau pour moi, celui de l'agriculture. Le projet Casdar Phenaufole porté conjointement par L'Institut Technique de la Betterave et IRSTEA se doit de répondre à des attentes sociétales et environnementales pour une agriculture éco-responsable.

Ce projet m'a permis d'approfondir mes connaissances sur l'utilisation de ROS et OpenCV. La conduite de mon projet de façon autonome, m'a amené à travailler à la fois sur l'environnement technique et logiciel me permettant ainsi de mieux appréhender l'ensemble des rouages, la prise en compte de l'existant tout en recherchant des solutions innovantes.

Différentes recherches bibliographiques pour la détection/segmentation de feuilles et le tracking ont été effectuées. Ces dernières m'ont amené à implémenter de nouvelles méthodes tout en acquérant de nouvelles connaissances très enrichissantes.

Pour mener à bien le projet dans son intégralité, j'envisage d'effectuer les tâches suivantes :

- Tester d'autres méthodes de tracking pour voir leurs limites,
- Terminer la machine à états pour le tracking,
- Entraîner le réseau Mask R-CNN avec les nouvelles images et vidéos obtenues lors des expérimentations effectuées sur le terrain,
- Assembler toutes les parties pour la machine à états finale (déplacement du robot, détection des feuilles, enregistrement des données, détection des maladies),
- Ajouter éventuellement une classe « centre » au réseau Mask R-CNN pour la détection des centres de betteraves,
- Ajouter un modèle de plante ressemblant à un plant de betterave pour remplacer le cylindre lors des simulations.

Ce projet a été une expérience particulièrement innovante et enrichissante ; il devrait me permettre par la suite de poursuivre des tests sur le terrain de façon à confronter les solutions apportées avec les attentes réelles, les exigences de chacun et les contraintes.

En outre, j'ai particulièrement apprécié le fait d'avoir eu une totale liberté dans mes recherches et dans ma façon de procéder ainsi qu'une écoute bienveillante lors de mes propositions de solutions.

## 12 Travail réalisé après la soutenance de stage

### Suite des recherches

Ces études [17], [50] présentent deux méthodes de tracking basées sur le contour de l'objet. Toutefois le temps de calcul reste important (environ 1 seconde par image).

L'un des paramètres important à prendre en compte le l'algorithme de tracking est l'invariance à l'échelle.

Il existe plusieurs études comparatives pour présenter différentes méthodes de tracking prenant en compte cet aspect [11], [37], [5].

Cette étude [8] présente une méthode utilisant des descripteurs de points en utilisant l'algorithme FAST [39].

Les auteurs de [29] présentent une implémentation du « Mean-shift algorithm » pour du tracking invariant à l'orientation et à l'échelle.

Cette étude [12] correspond au tracker DSST (Discriminative Scale Space Tracker) qui a remporté le challenge VOT 2014 (Visual Object Tracking) [28].

Pour la suite des travaux, nous avons utilisé le tracker DSST.

### 12.1 Suite des tests sur le tracking

Les algorithmes de tracking proposés par OpenCV permettent de retrouver l'objet à suivre sur plusieurs images successives en définissant au préalable l'encadrement de l'objet dans l'image initiale. Cependant, tous ces algorithmes ne sont pas invariant à l'échelle.

C'est-à-dire que la taille de l'encadrement défini dans l'image initiale restera fixe sur les images suivantes (Voir figure 14).

Il n'est donc pas possible de mettre une condition d'arrêt pas possible de mettre une condition d'arrêt quant à la distance d'approche de la feuille.

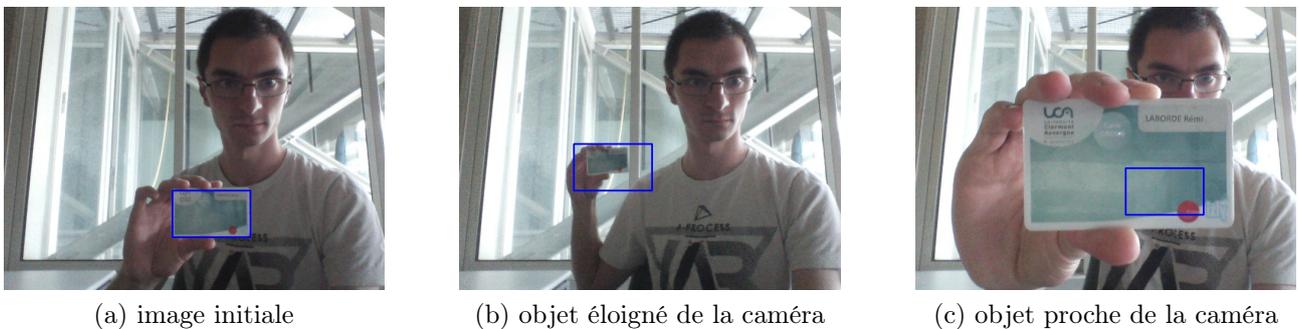


FIGURE 14 – Tests du tracker KCF [24]

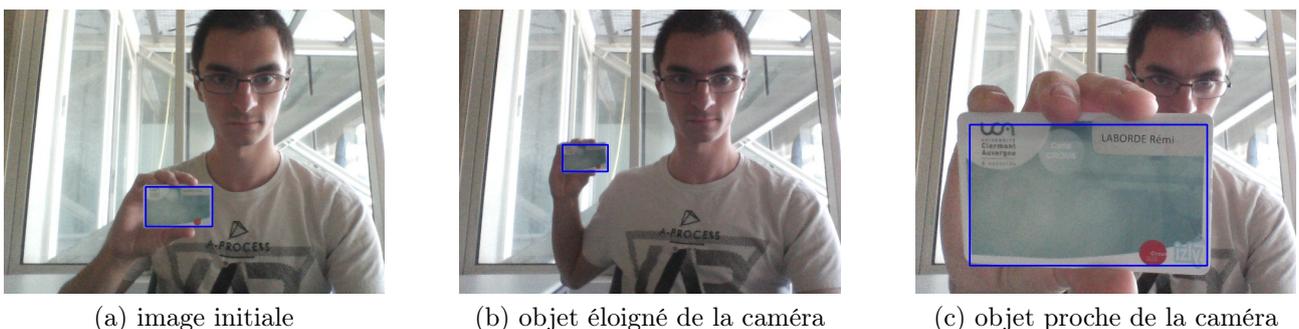


FIGURE 15 – Tests du tracker DSST [12]

## 12.2 Modifications de l'algorithme de détection lié à la commande

Toutes les modifications effectuées ci-après se trouvent dans le programme couplage.py.

$$\begin{cases} d_x = -\left(YG - \frac{H_{im}}{2}\right) \alpha \\ d_y = \left(XG - \frac{L_{im}}{2}\right) \alpha \end{cases}$$

$$\begin{cases} d_z = \left(R_{cercle}^d - radius\right) \alpha & \text{si } R < R_{zoom} \text{ et } limite\_gd == 0 \text{ et } limite\_hb == 0 \\ d_z = 0 & \text{sinon} \end{cases}$$

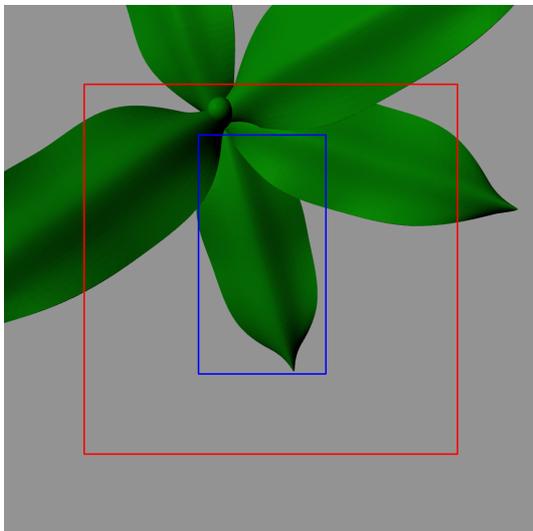
Le coefficient alpha était fixé à une valeur arbitraire tel que  $\alpha = \frac{1}{3000}$ .

$\alpha$  est maintenant fixé à  $\alpha = \frac{1}{3000+5*radius}$ .

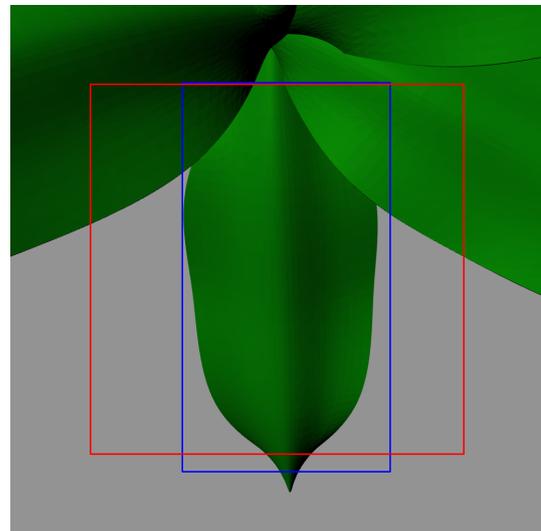
Le « radius » correspond à la longueur de la demi-diagonale du rectangle. On peut aussi voir cela comme le rayon du cercle circonscrit au rectangle.

La vitesse de déplacement pour la commande était uniquement proportionnelle à la distance entre le centre de la feuille et le centre de l'image. Elle l'est maintenant aussi proportionnelle à la taille de la feuille dans l'image. Plus un objet est proche de la caméra, plus le rectangle encadrant l'objet prend une place importante dans l'image, plus le coefficient  $\alpha$  est élevé et donc plus la vitesse de déplacement sera faible.

La variable « limite\_gd » correspond aux côtés gauche et droite du rectangle limite dans l'image. La variable « limite\_hb » correspond aux côtés haut et bas du rectangle limite dans l'image.



(a) Tracking en cours



(b) Feuille centrée et assez proche ( $limite\_hb = 1$ )

FIGURE 16 – Tracking de la feuille (en bleu) en fonction du rectangle limite (en rouge) pour 70% de la taille de l'image initiale

Si le rectangle encadrant la feuille, lorsqu'il est centré dans l'image, dépasse l'une de ces limites (haut et bas OU gauche et droite), on considère que la caméra est assez proche de la feuille.

La taille de ce rectangle limite est défini en fonction d'un pourcentage de la taille de l'image. Ce pourcentage peut être modifié dans le fichier config.yaml (variable : pourcentage\_zoom).

$R_{cercle}^d$  correspondait à la longueur du rayon désirée. Cette valeur était fixée dans le fichier `config.yaml`. Cette variable est maintenant égale à la longueur de la demi-diagonale du rectangle limite.

Une fois que la feuille est centrée dans l'image et assez proche de la caméra, on doit vérifier que la caméra est stabilisée au-dessus de la feuille, qu'il n'y ait pas d'oscillations. Pour cela, deux compteurs ont été mis en place. Par exemple, si dans les 20 prochaines images, la feuille est centrée dans l'image et que l'on a soit  $limite\_gd = 1$  soit  $limite\_hb = 1$ , on peut enregistrer l'image.

### **Modification du coefficient de pondération $e_y$** <sup>8</sup>

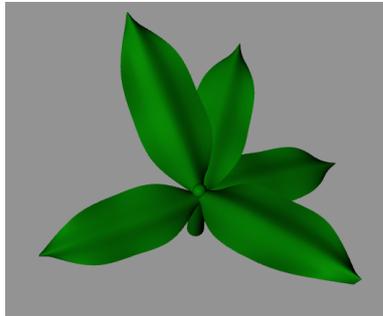
Dans le cas où le coefficient de pondération  $e_y = 0$ , cela signifie que le point à atteindre par l'extrémité du bras est inaccessible. Dans ce cas, le robot prend une photo la feuille dans sa position actuelle puis retourne à sa position initiale pour se diriger vers une autre feuille.

---

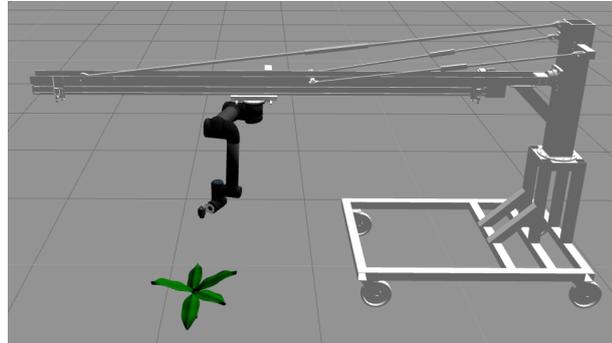
8. Pour plus de détails sur le coefficient de pondération  $e_y$ , voir le rapport final de thèse de Camille.

### 12.3 Plante artificielle pour la simulation et entraînement du réseau

Un modèle de plante a été ajouté. Il est maintenant possible d'utiliser le réseau Mask R-CNN pour faire la détection des feuilles puis d'utiliser les résultats de détection pour faire le tracking. Nous allons donc utiliser les rectangles obtenus en sortie du réseau.



(a) image de la plante modélisée



(b) image du robot dans l'environnement de simulation

FIGURE 17 – Simulation du système avec la plante modélisée

Le réseau a été entraîné à reconnaître les feuilles de cette plante. 16 images de ce plant ont été utilisées pour faire l'entraînement du réseau. On a donc un total de 80 feuilles. Pour la simulation il n'est pas nécessaire d'avoir une base de données importante puisque l'on ne travaille pas avec des conditions lumineuses variables.

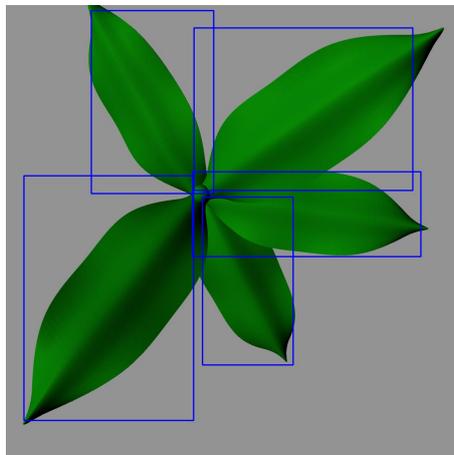


FIGURE 18 – Exemple de résultat de détection (chaque rectangle correspond à une feuille)

## 12.4 Machine à états du tracking

Voici le fonctionnement général de la machine à états :

### Etat n°0 : Vide-Buffer

- On passe à l'état n°1.

### Etat n°1 : Initialisation

- Déplacement du robot à la position initiale.
- Détection des feuilles dans l'image avec le réseau Mask-RCNN.
- Récupération des rectangles encadrant chacune des feuilles détectées.
- Initialisation du tracking.
- Initialisation de la première feuille à suivre à partir des informations récupérées.

### Etat n°2 : Tracking et déplacement du robot

- On récupère les coordonnées de la feuille à détecter dans l'image.
- Mise à jour de l'emplacement de la feuille dans l'image.
- Envoi des informations au programme couplage (topic).
- Si la caméra est au-dessus de la feuille (la feuille au centre de l'image), on prend une photo et on l'enregistre et on passe à l'état numéro 3.

### Etat n°3 : Déplacement du robot en position initiale

- Déplacement du robot à la position initiale.
- Si le robot est à la position initiale, on passe à l'état 4.

### Etat n°4 : Vide-Buffer

- On passe à l'état n°5.

### Etat n°5 : Passage à la feuille suivante ou retour à l'état initial

- S'il reste encore des feuilles à traiter, on passe à la feuille suivante et on passe à l'état n°2.
- Dans le cas contraire, la machine à états est terminée.

Les états n°0 et n°4 sont des états intermédiaires. Dans certains cas, il arrive que l'image retournée par le topic associé à la caméra ne corresponde pas à l'état dans lequel le robot se trouve actuellement. A la place l'image reçue correspond à l'état précédent (comme si la caméra avait une sorte de buffer qui n'était pas vidé).

Par exemple, lorsque le robot revient à la position initiale pour suivre une autre feuille (passage de l'état 3 à l'état 5), le tracker doit être réinitialisé pour suivre la feuille suivante.

Or, si l'image retournée par le topic n'est pas correcte, le tracker ne fonctionnera pas.

Dans ces deux états, aucun traitement n'est réalisé, on attend uniquement la prochaine image de la caméra qui, cette fois-ci, correspondra à l'état actuel du robot.

## 12.5 Travail restant

- Terminer la machine à états finale complète (détection des centres, détection des feuilles, détection des maladies)
- Augmenter la base de données pour améliorer la robustesse du réseau
- Tester les méthodes de détection comme YOLO (You Look Only Once) ou de type Faster-RCNN puisqu'on ne se sert pas des contours des feuilles donnés par le réseau. Cela permettrait de diminuer le temps de détection.

## 13 Annexes

### 13.1 Bibliographie

#### 13.1.1 Bibliographie pour la segmentation/détection des feuilles de betteraves

- [1] Waleed ABDULLA. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN). 2017.
- [2] R. ACHANTA et al. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012-11), p. 2274–2282. ISSN : 0162-8828, 2160-9292. DOI : [10.1109/TPAMI.2012.120](https://doi.org/10.1109/TPAMI.2012.120). URL : <http://ieeexplore.ieee.org/document/6205760/> (visité le 09/06/2019).
- [3] Shubhra AICH et Ian STAVNESS. “Leaf Counting with Deep Convolutional and Deconvolutional Networks”. In : *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017 IEEE International Conference on Computer Vision Workshop (ICCVW). Venice : IEEE, 2017-10, p. 2080–2089. ISBN : 978-1-5386-1034-3. DOI : [10.1109/ICCVW.2017.244](https://doi.org/10.1109/ICCVW.2017.244). URL : <http://ieeexplore.ieee.org/document/8265454/> (visité le 09/06/2019).
- [6] H G BARROW. “PARAMETRIC CORRESPONDENCE AND CHAMFER MATCHING : TWO NEW TECHNIQUES FOR IMAGE MATCHING”. In : (), p. 10.
- [9] Leo BREIMAN. “Random Forests”. In : *Machine Learning* 45.1 (2001-10-01), p. 5–32. ISSN : 1573-0565. DOI : [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324). URL : <https://doi.org/10.1023/A:1010933404324> (visité le 09/06/2019).
- [10] Liang-Chieh CHEN et al. “DeepLab : Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In : (2016-06-02). arXiv : [1606.00915 \[cs\]](https://arxiv.org/abs/1606.00915). URL : <http://arxiv.org/abs/1606.00915> (visité le 09/06/2019).
- [13] Bert DE BRABANDERE, Davy NEVEN et Luc VAN GOOL. “Semantic Instance Segmentation with a Discriminative Loss Function”. In : (2017-08-08). arXiv : [1708.02551 \[cs\]](https://arxiv.org/abs/1708.02551). URL : <http://arxiv.org/abs/1708.02551> (visité le 11/06/2019).
- [14] Andrei DOBRESCU, Mario Valerio GIUFFRIDA et Sotirios A TSAFTARIS. “Leveraging Multiple Datasets for Deep Leaf Counting”. In : (), p. 8.
- [15] Abhishek DUTTA et Andrew ZISSERMAN. “The VIA Annotation Software for Images, Audio and Video”. In : (2019-04-24). arXiv : [1904.10699 \[cs\]](https://arxiv.org/abs/1904.10699). URL : <http://arxiv.org/abs/1904.10699> (visité le 09/06/2019).
- [16] Mark EVERINGHAM et al. “The Pascal Visual Object Classes (VOC) Challenge”. In : *International Journal of Computer Vision* 88.2 (2010-06), p. 303–338. ISSN : 0920-5691, 1573-1405. DOI : [10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4). URL : <http://link.springer.com/10.1007/s11263-009-0275-4> (visité le 11/06/2019).
- [18] Alberto GARCIA-GARCIA et al. “A Review on Deep Learning Techniques Applied to Semantic Segmentation”. In : (2017-04-22). arXiv : [1704.06857 \[cs\]](https://arxiv.org/abs/1704.06857). URL : <http://arxiv.org/abs/1704.06857> (visité le 09/06/2019).
- [19] Ross GIRSHICK. “Fast R-CNN”. In : (2015-04-30). arXiv : [1504.08083 \[cs\]](https://arxiv.org/abs/1504.08083). URL : <http://arxiv.org/abs/1504.08083> (visité le 09/06/2019).

- [20] Ross GIRSHICK et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In : (2013-11-11). arXiv : [1311.2524 \[cs\]](https://arxiv.org/abs/1311.2524). URL : <http://arxiv.org/abs/1311.2524> (visité le 09/06/2019).
- [22] Esmael HAMUDA, Martin GLAVIN et Edward JONES. “A Survey of Image Processing Techniques for Plant Extraction and Segmentation in the Field”. In : *Computers and Electronics in Agriculture* 125 (2016-07), p. 184–199. ISSN : 01681699. DOI : [10.1016/j.compag.2016.04.024](https://doi.org/10.1016/j.compag.2016.04.024). URL : <https://linkinghub.elsevier.com/retrieve/pii/S0168169916301557> (visité le 09/06/2019).
- [23] Kaiming HE et al. “Mask R-CNN”. In : (2017-03-20). arXiv : [1703.06870 \[cs\]](https://arxiv.org/abs/1703.06870). URL : <http://arxiv.org/abs/1703.06870> (visité le 09/06/2019).
- [27] Kevin KELLER. “Soybean Leaf Coverage Estimation with Machine Learning and Thresholding Algorithms for Field Phenotyping”. In : (), p. 12.
- [30] Lei LI, Qin ZHANG et Danfeng HUANG. “A Review of Imaging Techniques for Plant Phenotyping”. In : *Sensors* 14.11 (2014-10-24), p. 20078–20111. ISSN : 1424-8220. DOI : [10.3390/s141120078](https://doi.org/10.3390/s141120078). URL : <http://www.mdpi.com/1424-8220/14/11/20078> (visité le 09/06/2019).
- [31] P. LOTTES et al. “An Effective Classification System for Separating Sugar Beets and Weeds for Precision Farming Applications”. In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden : IEEE, 2016-05, p. 5157–5163. ISBN : 978-1-4673-8026-3. DOI : [10.1109/ICRA.2016.7487720](https://doi.org/10.1109/ICRA.2016.7487720). URL : <http://ieeexplore.ieee.org/document/7487720/> (visité le 09/06/2019).
- [33] Massimo MINERVINI et al. “Finely-Grained Annotated Datasets for Image-Based Plant Phenotyping”. In : *Pattern Recognition Letters* 81 (2016-10), p. 80–89. ISSN : 01678655. DOI : [10.1016/j.patrec.2015.10.013](https://doi.org/10.1016/j.patrec.2015.10.013). URL : <https://linkinghub.elsevier.com/retrieve/pii/S0167865515003645> (visité le 09/06/2019).
- [34] Jean-Michel PAPE et Christian KLUKAS. “3-D Histogram-Based Segmentation and Leaf Detection for Rosette Plants”. In : *Computer Vision - ECCV 2014 Workshops*. Sous la dir. de Lourdes AGAPITO, Michael M. BRONSTEIN et Carsten ROTHER. T. 8928. Cham : Springer International Publishing, 2015, p. 61–74. ISBN : 978-3-319-16219-5 978-3-319-16220-1. DOI : [10.1007/978-3-319-16220-1\\_5](https://doi.org/10.1007/978-3-319-16220-1_5). URL : [http://link.springer.com/10.1007/978-3-319-16220-1\\_5](http://link.springer.com/10.1007/978-3-319-16220-1_5) (visité le 09/06/2019).
- [36] Tony PRIDMORE et al. “Computer Vision Problems in Plant Phenotyping, CVPPP 2018”. In : (2018), p. 4.
- [38] Shaoqing REN et al. “Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks”. In : (2015-06-04). arXiv : [1506.01497 \[cs\]](https://arxiv.org/abs/1506.01497). URL : <http://arxiv.org/abs/1506.01497> (visité le 09/06/2019).
- [40] Bryan C. RUSSELL et al. “LabelMe : A Database and Web-Based Tool for Image Annotation”. In : *International Journal of Computer Vision* 77.1-3 (2008-05), p. 157–173. ISSN : 0920-5691, 1573-1405. DOI : [10.1007/s11263-007-0090-8](https://doi.org/10.1007/s11263-007-0090-8). URL : <http://link.springer.com/10.1007/s11263-007-0090-8> (visité le 09/06/2019).
- [41] Amaia SALVADOR et al. “Recurrent Neural Networks for Semantic Instance Segmentation”. In : (2017-12-02). arXiv : [1712.00617 \[cs\]](https://arxiv.org/abs/1712.00617). URL : <http://arxiv.org/abs/1712.00617> (visité le 09/06/2019).
- [42] Hanno SCHARR et al. “Annotated Image Datasets of Rosette Plants”. In : (), p. 16.

- [43] Hanno SCHARR et al. “Leaf Segmentation in Plant Phenotyping : A Collation Study”. In : *Machine Vision and Applications* 27.4 (2016-05), p. 585–606. ISSN : 0932-8092, 1432-1769. DOI : [10.1007/s00138-015-0737-3](https://doi.org/10.1007/s00138-015-0737-3). URL : <http://link.springer.com/10.1007/s00138-015-0737-3> (visité le 09/06/2019).
- [47] Miss Hetal J VALA et Astha BAXI. “A Review on Otsu Image Segmentation Algorithm”. In : 2.2 (), p. 3.
- [48] L. VINCENT et P. SOILLE. “Watersheds in Digital Spaces : An Efficient Algorithm Based on Immersion Simulations”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6 (1991-06), p. 583–598. ISSN : 01628828. DOI : [10.1109/34.87344](https://doi.org/10.1109/34.87344). URL : <http://ieeexplore.ieee.org/document/87344/> (visité le 09/06/2019).
- [49] Daniel WARD, Peyman MOGHADAM et Nicolas HUDSON. “Deep Leaf Segmentation Using Synthetic Data”. In : (2018-07-28). arXiv : [1807.10931](https://arxiv.org/abs/1807.10931) [cs]. URL : <http://arxiv.org/abs/1807.10931> (visité le 09/06/2019).
- [51] Xi YIN et al. “Joint Multi-Leaf Segmentation, Alignment, and Tracking for Fluorescence Plant Videos”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018-06-01), p. 1411–1423. ISSN : 0162-8828, 2160-9292. DOI : [10.1109/TPAMI.2017.2728065](https://doi.org/10.1109/TPAMI.2017.2728065). URL : <https://ieeexplore.ieee.org/document/7982753/> (visité le 09/06/2019).
- [52] Yezi ZHU. “Data Augmentation Using Conditional Generative Adversarial Networks for Leaf Counting in Arabidopsis Plants”. In : (), p. 11.

### 13.1.2 Bibliographie pour le tracking

- [4] Boris BABENKO, Ming-Hsuan YANG et Serge BELONGIE. “Visual Tracking with Online Multiple Instance Learning”. In : (), p. 8.
- [5] R.V. BABU, P. PEREZ et P. BOUTHEMY. “Robust Tracking with Motion Estimation and Kernel-Based Color Modelling”. In : *IEEE International Conference on Image Processing 2005*. 2005 International Conference on Image Processing. Genova, Italy : IEEE, 2005, p. I-717. ISBN : 978-0-7803-9134-5. DOI : [10.1109/ICIP.2005.1529851](https://doi.org/10.1109/ICIP.2005.1529851). URL : <http://ieeexplore.ieee.org/document/1529851/> (visité le 13/08/2019).
- [8] P. BILINSKI, F. BREMOND et M.B. KAANICHE. “Multiple Object Tracking with Occlusions Using HOG Descriptors and Multi Resolution Images”. In : *3rd International Conference on Imaging for Crime Detection and Prevention (ICDP 2009)*. 3rd International Conference on Imaging for Crime Detection and Prevention (ICDP 2009). London, UK : IET, 2009, P36-P36. ISBN : 978-1-84919-207-1. DOI : [10.1049/ic.2009.0264](https://doi.org/10.1049/ic.2009.0264). URL : <https://digital-library.theiet.org/content/conferences/10.1049/ic.2009.0264> (visité le 13/08/2019).
- [11] Martin DANELLJAN et al. “Accurate Scale Estimation for Robust Visual Tracking”. In : *Proceedings of the British Machine Vision Conference 2014*. British Machine Vision Conference 2014. Nottingham : British Machine Vision Association, 2014, p. 65.1-65.11. ISBN : 978-1-901725-52-0. DOI : [10.5244/C.28.65](https://doi.org/10.5244/C.28.65). URL : <http://www.bmva.org/bmvc/2014/papers/paper038/index.html> (visité le 13/08/2019).
- [12] Martin DANELLJAN et al. “Discriminative Scale Space Tracking”. In : (20 sept. 2016). arXiv : [1609.06141 \[cs\]](https://arxiv.org/abs/1609.06141). URL : <http://arxiv.org/abs/1609.06141> (visité le 13/08/2019).
- [17] Hua FANG, Jeong-Woo KIM et Jong-Whan JANG. “A Fast Snake Algorithm for Tracking Multiple Objects”. In : *Journal of Information Processing Systems 7.3* (30 sept. 2011), p. 519-530. ISSN : 1976-913X. DOI : [10.3745/JIPS.2011.7.3.519](https://doi.org/10.3745/JIPS.2011.7.3.519). URL : <http://koreascience.or.kr/journal/view.jsp?kj=E1JBB0&py=2011&vnc=v7n3&sp=519> (visité le 13/08/2019).
- [21] H. GRABNER, M. GRABNER et H. BISCHOF. “Real-Time Tracking via On-Line Boosting”. In : *Proceedings of the British Machine Vision Conference 2006*. British Machine Vision Conference 2006. Edinburgh : British Machine Vision Association, 2006, p. 6.1-6.10. ISBN : 978-1-901725-32-2. DOI : [10.5244/C.20.6](https://doi.org/10.5244/C.20.6). URL : <http://www.bmva.org/bmvc/2006/papers/033.html> (visité le 23/06/2019).
- [24] João F. HENRIQUES et al. “High-Speed Tracking with Kernelized Correlation Filters”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence 37.3* (1<sup>er</sup> mar. 2015), p. 583-596. ISSN : 0162-8828, 2160-9292. DOI : [10.1109/TPAMI.2014.2345390](https://doi.org/10.1109/TPAMI.2014.2345390). arXiv : [1404.7584](https://arxiv.org/abs/1404.7584). URL : <http://arxiv.org/abs/1404.7584> (visité le 23/06/2019).
- [25] Zdenek KALAL, Krystian MIKOLAJCZYK et Jiri MATAS. “Forward-Backward Error : Automatic Detection of Tracking Failures”. In : *2010 20th International Conference on Pattern Recognition*. 2010 20th International Conference on Pattern Recognition (ICPR). Istanbul, Turkey : IEEE, août 2010, p. 2756-2759. ISBN : 978-1-4244-7542-1. DOI : [10.1109/ICPR.2010.675](https://doi.org/10.1109/ICPR.2010.675). URL : <http://ieeexplore.ieee.org/document/5596017/> (visité le 23/06/2019).

- [26] Zdenek KALAL, Krystian MIKOLAJCZYK et Jiri MATAS. “Tracking-Learning-Detection”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (juil. 2012), p. 1409–1422. ISSN : 0162-8828, 2160-9292. DOI : [10.1109/TPAMI.2011.239](https://doi.org/10.1109/TPAMI.2011.239). URL : <http://ieeexplore.ieee.org/document/6104061/> (visité le 23/06/2019).
- [28] Matej KRISTAN et al. “The Visual Object Tracking VOT2014 Challenge Results”. In : (), p. 27.
- [29] KWANG MOO YI, HO SEOK AHN et JIN YOUNG CHOI. “Orientation and Scale Invariant Mean Shift Using Object Mask-Based Kernel”. In : *2008 19th International Conference on Pattern Recognition*. 2008 19th International Conference on Pattern Recognition (ICPR). Tampa, FL, USA : IEEE, déc. 2008, p. 1–4. ISBN : 978-1-4244-2174-9. DOI : [10.1109/ICPR.2008.4761156](https://doi.org/10.1109/ICPR.2008.4761156). URL : <http://ieeexplore.ieee.org/document/4761156/> (visité le 13/08/2019).
- [35] Sarika PATEL, Udesang K JALIYA et Mahashweta J JOSHI. “A Review : Object Detection and Object Tracking Methods”. In : *International Journal of Innovative and Emerging Research in Engineering* 2.1 (2015), p. 6.
- [37] Karan RAMPAL, Kazuyuki SAKURAI et Hitoshi IMAOKA. “Fast and Accurate Scale Estimation Method for Object Tracking”. In : *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016 23rd International Conference on Pattern Recognition (ICPR). Cancun : IEEE, déc. 2016, p. 2712–2715. ISBN : 978-1-5090-4847-2. DOI : [10.1109/ICPR.2016.7900045](https://doi.org/10.1109/ICPR.2016.7900045). URL : <http://ieeexplore.ieee.org/document/7900045/> (visité le 13/08/2019).
- [39] Edward ROSTEN et Tom DRUMMOND. “Machine Learning for High-Speed Corner Detection”. In : *Computer Vision – ECCV 2006*. Sous la dir. d’Aleš LEONARDIS, Horst BISCHOF et Axel PINZ. T. 3951. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 430–443. ISBN : 978-3-540-33832-1 978-3-540-33833-8. DOI : [10.1007/11744023\\_34](https://doi.org/10.1007/11744023_34). URL : [http://link.springer.com/10.1007/11744023\\_34](http://link.springer.com/10.1007/11744023_34) (visité le 13/08/2019).
- [44] Akanksha SHARMA et Dr Deepak DEMBLA. “Object Detection and Tracking Techniques : A Review”. In : 5.6 (2007), p. 12.
- [45] Mukesh TIWARI. “A Review of Detection and Tracking of Object from Image and Video Sequences”. In : (), p. 22.
- [46] Dhara TRAMBADIYA et Chintan VARNAGAR. “A Review on Moving Object Detection and Tracking Methods”. In : *International Journal of Advance Engineering and Research Development* (), p. 9.
- [50] N. XU et N. AHUJA. “Object Contour Tracking Using Graph Cuts Based Active Contours”. In : *Proceedings. International Conference on Image Processing*. IEEE International Geoscience and Remote Sensing Symposium. IGARSS 2002. T. 1. Toronto, Ont., Canada : IEEE, 2002, p. III–277–III–280. ISBN : 978-0-7803-7622-9. DOI : [10.1109/ICIP.2002.1038959](https://doi.org/10.1109/ICIP.2002.1038959). URL : <http://ieeexplore.ieee.org/document/1038959/> (visité le 13/08/2019).

## 13.2 Principales fonctions d'OpenCV utilisées

Voici une liste exhaustive des fonctions d'openCV utilisées dans les programmes existants de détection de feuilles et de centres de betteraves.

- bitwise\_not  
[https://docs.opencv.org/2.4/modules/core/doc/operations\\_on\\_arrays.html#bitwise-not](https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#bitwise-not)
- canny  
[https://docs.opencv.org/2.4/modules/imgproc/doc/feature\\_detection.html?highlight=canny#canny](https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=canny#canny)
- circle  
[https://docs.opencv.org/2.4/modules/core/doc/drawing\\_functions.html#circle](https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html#circle)
- ConvertTo  
[http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/core/doc/basic\\_structures.html#mat-convertto](http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/core/doc/basic_structures.html#mat-convertto)
- CopyTo  
[http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/core/doc/basic\\_structures.html#mat-copyto](http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/core/doc/basic_structures.html#mat-copyto)
- connectedComponentsWithStats  
[https://docs.opencv.org/3.1.0/d3/dc0/group\\_\\_imgproc\\_\\_shape.html#gac2718a64ade63475](https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#gac2718a64ade63475)  
[https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural\\_analysis\\_and\\_shape\\_descriptors.html?highlight=connectedcomponents#connectedcomponents](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=connectedcomponents#connectedcomponents)
- cvtColor  
[https://docs.opencv.org/2.4.13.7/modules/imgproc/doc/miscellaneous\\_transformations.html#cvtColor](https://docs.opencv.org/2.4.13.7/modules/imgproc/doc/miscellaneous_transformations.html#cvtColor)
- dilate  
<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#dilate>
- distanceTransform  
[https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous\\_transformations.html?highlight=watershed#distancetransform](https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=watershed#distancetransform) [https://www.tutorialspoint.com/opencv/opencv\\_distance\\_transformation.htm](https://www.tutorialspoint.com/opencv/opencv_distance_transformation.htm)
- drawContours  
[https://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing\\_functions.html#drawcontours](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing_functions.html#drawcontours)
- erode  
<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#erode>
- findContours  
[https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural\\_analysis\\_and\\_shape\\_descriptors.html?highlight=connectedcomponents#findcontours](https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=connectedcomponents#findcontours)
- GaussianBlur  
<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#gaussianblur>
- getStructuringElement  
<https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#getstructuringelement>
- HoughlinesP  
[https://docs.opencv.org/2.4/modules/imgproc/doc/feature\\_detection.html?highlight=houghlinesp#houghlinesp](https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghlinesp#houghlinesp)

- `inRange`  
[https://docs.opencv.org/3.0-beta/modules/core/doc/operations\\_on\\_arrays.html?highlight=inrange#inrange](https://docs.opencv.org/3.0-beta/modules/core/doc/operations_on_arrays.html?highlight=inrange#inrange)
- `imread`  
[https://docs.opencv.org/3.0-alpha/modules/imgcodecs/doc/reading\\_and\\_writing\\_images.html#imread](https://docs.opencv.org/3.0-alpha/modules/imgcodecs/doc/reading_and_writing_images.html#imread)
- `imwrite`  
[https://docs.opencv.org/3.0-alpha/modules/imgcodecs/doc/reading\\_and\\_writing\\_images.html#imwrite](https://docs.opencv.org/3.0-alpha/modules/imgcodecs/doc/reading_and_writing_images.html#imwrite)
- `normalize`  
[https://docs.opencv.org/2.4/modules/core/doc/operations\\_on\\_arrays.html?#normalize](https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html?#normalize)
- `Scalar`  
[https://docs.opencv.org/2.4/doc/tutorials/core/basic\\_geometric\\_drawing/basic\\_geometric\\_drawing.html#scalar](https://docs.opencv.org/2.4/doc/tutorials/core/basic_geometric_drawing/basic_geometric_drawing.html#scalar)
- `treshold`  
[https://docs.opencv.org/3.4/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html)
- `watershed`  
[https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous\\_transformations.html?highlight=watershed#watershed](https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=watershed#watershed)  
<https://www.pyimagesearch.com/2015/11/02/watershed-opencv/>

### 13.3 Explication du calcul de la précision moyenne de la partie 9.2.3

Dans cette section, nous allons expliquer comment est calculée la précision moyenne<sup>9</sup> d'un réseau. Prenons l'exemple d'une image où 5 feuilles sont annotées. Dans cette image le réseau a trouvé 4 feuilles. On va regarder tous les contours prédits du réseau ayant une intersection avec chaque contour annoté.

Pour cela, on calcule l'IoU (Intersection over Union) entre chaque contour annoté et chaque contour prédit. l'IoU est le ratio entre l'intersection des deux aires sur l'union des deux aires. Plus le résultat est proche de 0, plus le contour prédit est mauvais et inversement si le résultat est proche de 1.

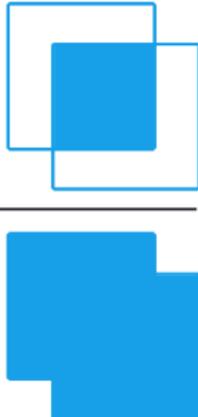
$$\text{IoU} = \frac{\text{Aire de l'intersection}}{\text{Aire de l'union}}$$


FIGURE 19 – Formule pour le calcul de l'IoU

Dans notre cas, nous aurons un tableau de 5X5 comprenant tous les résultats des IoU entre chaque contour prédit et chaque contour annoté.

On garde les meilleurs résultats pour chaque contour annoté. On obtient donc un tableau 5X1 contenant le meilleur résultat de l'IoU pour un contour annoté par rapport à un contour prédit.

Dans notre exemple, on peut avoir un résultat comme celui-ci :

resultats\_IoU : [0.64 ; 0.28 ; 0.72 ; 0.91 ; 0.46]

On les classe du meilleur résultat au plus mauvais.

on obtient : resultats\_IoU : [0.91 ; 0.72 ; 0.64 ; 0.46 ; 0.28]

Ensuite, les résultats ayant un IoU supérieur à 0.5 sont mis à 1 ; les autres résultats sont mis à 0. C'est un seuil permettant de considérer si une feuille est bien détectée par le réseau ou non.

On obtient : resultats\_IoU : [1 ; 1 ; 1 ; 0 ; 0]

A partir de cela, on peut calculer la « précision » et le « recall ». La précision : c'est le ratio entre le nombre de résultats prédits ayant une IoU supérieur à 0.5 sur le nombre de résultats prédits.

Dans notre cas, le résultat prédit 4 feuilles, donc la précision comportera 4 résultats.

resultats\_IoU : [1 ; 1 ; 1 ; 0 ; 0]

nombre de feuilles : [1 ; 2 ; 3 ; 4 ;]

précision : [ $\frac{1}{1}$  ;  $\frac{2}{2}$  ;  $\frac{3}{3}$  ;  $\frac{3}{4}$ ]

ce qui équivaut à : précision : [1 ; 1 ; 1 ;  $\frac{3}{4}$ ]

9. La précision moyenne est souvent notée AP (Average Precision).

Le recall : c'est le ratio entre le nombre de résultats prédits ayant une IoU supérieur à 0.5 sur le nombre de résultats annotés.

Dans notre cas, nous avons 5 feuilles annotées :

On a : [1 ; 1 ; 1 ; 0 ; 0]

recall : [ $\frac{1}{5}$  ;  $\frac{2}{5}$  ;  $\frac{3}{5}$  ;  $\frac{3}{5}$ ]

Enfin, on trace la courbe « precision-recall » en ayant pour abscisse le recall et pour ordonnée la précision. La précision moyenne correspond à l'intégrale de la courbe obtenue. Mais pour obtenir un résultat compris entre 0 et 1, les abscisses et ordonnées [0;1] et [1;1] sont ajoutées.

On a donc :

precision : [1 ; 1 ; 1 ; 1 ;  $\frac{3}{4}$  ; 0]

recall : [0 ;  $\frac{1}{5}$  ;  $\frac{2}{5}$  ;  $\frac{3}{5}$  ;  $\frac{3}{5}$  ; 1]

L'intégrale est donc calculée après l'ajout de ces points.

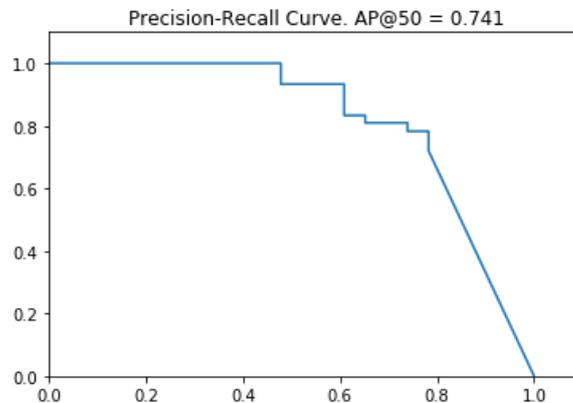


FIGURE 20 – Exemple de courbe Precision-Recall

On a donc la précision moyenne pour une image. Dans le cadre du rapport, il y a donc 10 images avec leur précision moyenne respective, Une simple moyenne entre ses 10 « précisions moyennes » est réalisée pour obtenir le résultat de 0.58.

Pour plus d'informations sur le calcul de la précision moyenne et de l'IoU :

[https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121)

<https://www.jeremyjordan.me/evaluating-image-segmentation-models/>

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

<https://sanchom.wordpress.com/tag/average-precision/>

## 13.4 Architecture logicielle du projet ROS

Les différents paquets ROS et Tensorflow nécessaires pour la mise en place du projet ont été installés sur un nouvel ordinateur. De plus, le projet initial qui utilisait la version ROS Indigo a été implémenté sous ROS Kinetic. Plusieurs modifications du projet ont dû être effectuées pour s'adapter à la nouvelle version.

### 13.4.1 Architecture des packages ROS

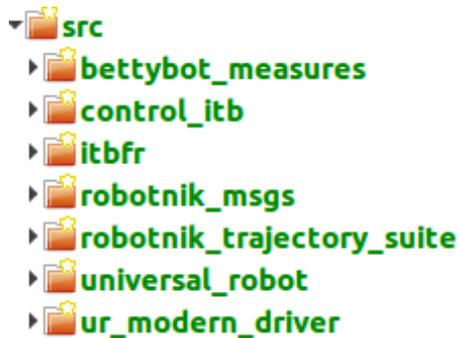


FIGURE 21 – Architecture du projet ROS

#### Package `bettybot_measures`

Contient tous les algorithmes de traitement d'images pour le travail existant, que ce soit pour la détection des centres des plants de betteraves ou la détection des feuilles.

Dans le dossier `launch` :

- `detectbeetnouv.launch` : Permet de lancer le programme « `detect_leaf.cpp` » ou « `detect_center.cpp` » (À modifier dans le fichier `detectbeetnouv.launch`).

Dans le dossier `scripts` :

- `detect_leaf.cpp` : Programme pour détecter les feuilles de betteraves
- `detect_center.cpp` : Programme pour détecter les centres des plants de betteraves

#### Package `control_itb`

Contient tous les algorithmes de commande pour le travail existant, que ce soit pour la détection du cylindre ou la commande complète du robot.

Dans le dossier `scripts` : Contient tous les programmes python pour la commande.

Cinq fichiers importants :

- `detecter.py` : Programme pour détecter le cylindre vert et déterminer sa position
- `detecter_final.py` : Programme comprenant la machine à états du tracking. (Détection des feuilles avec le Mask R-CNN puis déplacement au-dessus de chacune d'elles)
- `couplage.py` : Programme faisant la liaison entre la vision et la commande (`detecter.py/detecter_final.py` et `plan_execute.py`)
- `plan_execute.py` : Programme permettant de contrôler les 7 degrés de liberté du robot (axe linéaire + bras UR5)
- `GestionActionsBet.py` : Programme comprenant la machine à états globale du projet (Détection des centres/feuilles, Contrôle/commande du robot, Détection des maladies)

Pour plus d'informations sur les commandes à effectuer pour faire les tests en simulation, lire le fichier `readme.md`

## Dossier itbfr

En résumé, tous les fichiers et packages présents dans celui-ci permettent de mettre en place l'environnement de simulation Gazebo pour simuler le fonctionnement du robot complet (axe linéaire + bras UR5).

Un fichier .launch est utilisé pour faire apparaître le robot dans l'environnement de simulation Gazebo et Rviz : itbfr\_sim\_bringup.launch.

Il se trouve dans le dossier itbfr/itbfr\_common/itbfr\_bringup/launch.

## Packages :

**robotnik\_msgs, robotnik\_trajectory\_suite, universal\_robot, ur\_modern\_driver**

Tous ces packages ont été produits par la société Robotnik et servent au bon fonctionnement du projet.

### 13.4.2 Librairies complémentaires

Comme dit précédemment dans la partie Environnement logiciel, il est nécessaire d'installer les packages suivants : tensorflow\_ros\_cpp, tensorflow\_catkin et catkin simple. Ils sont utilisés pour faire fonctionner les algorithmes de détection de centres et de feuilles du travail existant.

### 13.4.3 Fichiers/Dossiers complémentaires

Dans le dossier beet\_photos se trouvent 3 dossiers contenant les images utilisées pour l'entraînement du réseau Inception V3.

Dans le dossier beet\_photos

- dossier beet\_background : Contient environ 200 images comportant entre autre de la terre ou des cailloux.
- dossier beet\_center : Contient environ 200 images de centre de plants de betteraves
- dossier beet\_feuilles : Contient environ 200 images de feuilles de betteraves

7 fichiers .bag ont été utilisés pour faire des tests ou pour entraîner le réseau Mask R-CNN.

4 fichiers .bag sont des vidéos prises par le robot Phenaufof dans le hall technologique d'IRSTEA. Ces vidéos ont été utilisées pour tester les algorithmes existants pour la détection de centre et de feuilles de betteraves.

Dans le dossier bagfiles :

- ficA-2018-06-28-09-29-57.bag
- ficB-2018-06-28-09-32-00.bag
- ficC-2018-06-28-09-47-17.bag
- ficD-2018-06-28-09-48-05.bag

Les 3 autres fichiers .bag sont des vidéos prises par le robot RobuFast en champs. Ces vidéos ont été utilisées pour l'entraînement du réseau Mask R-CNN.

Dans le dossier bagfiles :

- OpeRose\_2018-06-27-13-41-21.bag
- OpeRose\_2018-06-27-14-02-52.bag
- OpeRose\_2018-06-27-14-15-35.bag

## 13.5 Architecture du dossier Mask R-CNN

Les dossiers et fichiers les plus utilisés sont présentés ici. Les programmes utilisés pour la détection de feuilles seront ajoutés dans le projet ROS par la suite.

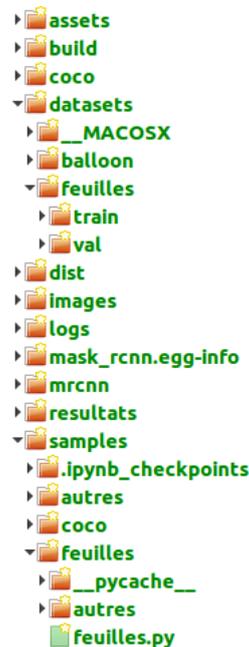


FIGURE 22 – Architecture du dossier Mask R-CNN

### Dossier datasets

Contient les dossiers et les images ayant servi pour l'entraînement du réseau et son évaluation.

Dans le dossier feuilles :

- dossier train : Contient les 20 images utilisées pour l'entraînement ainsi que le fichier .json comprenant les annotations de ces 20 images
- dossier val : Contient les 10 images pour l'évaluation ainsi que le fichier .json comprenant les annotations de ces 10 images

### Dossier samples

Dans le dossier coco :

- inspect\_model.ipynb : Fichier de type jupyter-notebook utilisé pour évaluer le réseau à partir du dossier de validation.

Dans le dossier feuilles :

- feuilles.py : Programme utilisé pour l'entraînement du réseau et pour tester le réseau sur différentes images

L'entraînement du réseau en utilisant les poids initiaux de la base COCO se fait via la commande suivante :

```
python3 feuilles.py train -dataset=/home/.../datasets/feuilles -weights=COCO
```

Si l'on veut entraîner le réseau avec les derniers poids obtenus (du dossier logs) :

```
python3 feuilles.py train -dataset=/home/.../datasets/feuilles -weights=last
```

Une fonction a été réalisée pour tester le réseau sur différentes images. Dans ce cas, on utilise la commande suivante :

```
python3 feuilles.py test --weights=last
```

### Dossier logs

C'est dans ce dossier que le réseau donne les fichiers .h5 contenant les poids du réseau Mask R-CNN après l'entraînement. L'entraînement va donner un fichier .h5 à la fin de chaque rétro-propagation.

Les tests fonctionnent en python 2.7 et ont pu être implémentés dans le projet ROS. L'entraînement est fait avec python 3.5. Je n'ai pas réussi à effectuer les modifications pour que la partie « Entraînement » fonctionne sous python 2.7.

## 13.6 Tutoriel pour réaliser l'entraînement du réseau sur de nouvelles images

- Créer une base d'images pour l'entraînement et la diviser en deux groupes : un pour l'entraînement (dossier train : environ 80% des images) et l'autre pour la validation (dossier val : environ 20% des images).
- Segmenter les feuilles pour chaque groupe avec l'outil VGG Annotator ([15] que je vous invite à lire pour plus de détails). Chaque dossier (train et val) aura son propre fichier json.
- Récupérer le fichier .json résultant via l'onglet Annotations → export Annotations as json
- Mettre dans le dossier train et val les deux fichiers .json correspondant
- Dans un terminal, se placer dans le dossier où le programme feuilles.py est présent
- Entraîner le réseau en spécifiant le bon chemin vers la base de données

### Segmenter les images avec VGG Annotator<sup>10</sup>

- Ajouter les images via « Add Files »
- Choisir « Polygon Region Shape » dans la section Region Shape
- Segmenter une feuille point par point (au clic) puis appuyer sur « Entrée » pour valider la segmentation

Il est possible de définir des catégories dans la section Attributes puis region Attributes.

On peut créer deux catégories, (par exemple centres et feuilles) pour ensuite associer chaque segmentation à une catégorie.

Dans le cas où il n'y a qu'un seul type d'objet à segmenter (dans notre cas les feuilles), définir des catégories n'est pas nécessaire puisqu'il n'y aura qu'une seule classe par défaut dans le réseau.

Il est possible de sauvegarder les annotations et les images sous forme d'un projet avec :

Project → Save

Cela va créer un fichier .json. On peut charger un projet en cours avec :

Project → Load.

Il sera nécessaire d'ajouter à nouveau les photos avec Add Files car elles ne seront probablement pas reconnues. Une fois les images chargées les annotations effectuées précédemment réapparaîtront.

---

10. [http://www.robots.ox.ac.uk/~vgg/software/via/via\\_demo.html](http://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html)

## 13.7 Déroulement du projet

### 13.7.1 Diagramme de Gant

Tâches	18 - 24	25 - 3	4 - 10	11 - 17	18 - 24	25 - 31	1 - 7	8 - 14	15 - 21
	Fev	Fev/Mar	Mars	Mars	Mars	Mars	Avril	Avril	Avril
<b>Environnement de travail</b>									
Mise en place du projet sous ROS Kinetic									
Tests des fonctions existantes									
Analyse du travail existant									
<b>Bibliographie</b>									
Recherches Détection/segmentation des feuilles									
Recherches Tracking									
Analyse et choix des solutions retenues									
<b>Programmation perception active</b>									
Fonction détection des centres de betteraves									
Fonction détection des feuilles									
Implémentation du réseau Mask R-CNN									
Fonction Tracking / commande bras									
Mise en place de la machine à états finale									
<b>Tests et simulations</b>									
Simulation des fonctions réalisées sous Gazebo									
Tests des fonctions en champ									

FIGURE 23 – Diagramme de Gant du 18 février au 21 avril

Tâches	22 - 28	29 - 5	6 - 12	13 - 19	20 - 26	27 - 2	3 - 9	10 - 16	17 - 23
	Avril	Avr/Mai	Mai	Mai	Mai	Mai/Juin	Juin	Juin	Juin
<b>Environnement de travail</b>									
Mise en place du projet sous ROS Kinetic									
Tests des fonctions existantes									
Analyse du travail existant									
<b>Bibliographie</b>									
Recherches Détection/segmentation des feuilles									
Recherches Tracking									
Analyse et choix des solutions retenues									
<b>Programmation perception active</b>									
Fonction détection des centres de betteraves									
Fonction détection des feuilles									
Implémentation du réseau Mask R-CNN									
Fonction Tracking / commande bras									
Mise en place de la machine à états finale									
<b>Tests et simulations</b>									
Simulation des fonctions réalisées sous Gazebo									
Tests des fonctions en champ									

FIGURE 24 – Diagramme de Gant du 22 avril au 23 juin

Tâches	24 - 30	1 - 7	8 - 14	15 - 21	22 - 28	29 - 4	5 - 11	12 - 18
	Juin	Juillet	Juillet	Juillet	Juillet	Jui/Aou	Août	Août
<b>Environnement de travail</b>								
Mise en place du projet sous ROS Kinetic								
Tests des fonctions existantes								
Analyse du travail existant								
<b>Bibliographie</b>								
Recherches Détection/segmentation des feuilles								
Recherches Tracking								
Analyse et choix des solutions retenues								
<b>Programmation perception active</b>								
Fonction détection des centres de betteraves								
Fonction détection des feuilles								
Implémentation du réseau Mask R-CNN								
Fonction Tracking / commande bras								
Mise en place de la machine à états finale								
<b>Tests et simulations</b>								
Simulation des fonctions réalisées sous Gazebo								
Tests des fonctions en champ								

FIGURE 25 – Diagramme de Gant du 24 juin au 18 août

### 13.7.2 Problèmes rencontrés

Différents problèmes ont été rencontrés lors de la mise en place de l'environnement de travail :

- Le changement de version de ROS indigo à ROS kinetic a généré beaucoup d'anomalies de version et en conséquence a demandé un investissement non négligeable pour les corrections de ces anomalies.
- Les dernières versions des codes réalisés pour ce projet n'avaient pas été mis à jour sur la plateforme GitLab prévue à cet effet ce qui m'a contraint à repartir d'une ancienne version issue la forge « GitLab » puis y ajouter après coup les derniers fichiers via une clé USB.
- Déterminer quels étaient les packages nécessaires à télécharger afin d'avoir le projet complet fonctionnel surtout pour l'utilisation Tensorflow.
- Beaucoup de difficultés également pour reconstituer et comprendre l'ensemble du travail réalisé, que ce soit d'une part par la complexité des algorithmes mis en place, le manque d'explications détaillées concernant leur fonctionnement (y compris dans les rapports faits précédemment) et d'autre part, par l'absence de tout commentaire dans les codes sources.

## 13.8 Liste des figures

### Table des figures

1	Représentations du robot Phenaufol . . . . .	6
2	Photo du robot Phenaufol . . . . .	7
3	Fonctionnement de l'algorithme . . . . .	17
4	Résultats de l'algorithme de détection du cylindre . . . . .	17
5	Décomposition de l'image . . . . .	18
6	Exemples d'images des bases de données A1, A2, A3 (images tirées de [42]) . . . . .	20
7	Architecture du réseau Mask-RCNN(image tirée de [23]) . . . . .	22
8	Exemples d'images tirées des différentes vidéos . . . . .	23
9	Exemple d'une image segmentée avec VGG Annotator et utilisée pour l'entraî- nement . . . . .	23
10	Exemple de résultat du réseau Mask R-CNN . . . . .	25
11	Les différentes catégories de suivi d'objets (image tirée de [35]) . . . . .	26
12	Image du robot dans l'environnement de simulation Gazebo . . . . .	28
13	Visualisation des différents états possibles . . . . .	29
14	Tests du tracker KCF [24] . . . . .	31
15	Tests du tracker DSST [12] . . . . .	31
16	Tracking de la feuille (en bleu) en fonction du rectangle limite (en rouge) pour 70% de la taille de l'image initiale . . . . .	32
17	Simulation du système avec la plante modélisée . . . . .	34
18	Exemple de résultat de détection (chaque rectangle correspond à une feuille) . . . . .	34
19	Formule pour le calcul de l'IoU . . . . .	43
20	Exemple de courbe Precision-Recall . . . . .	44
21	Architecture du projet ROS . . . . .	45
22	Architecture du dossier Mask R-CNN . . . . .	47
23	Diagramme de Gant du 18 février au 21 avril . . . . .	49
24	Diagramme de Gant du 22 avril au 23 juin . . . . .	49
25	Diagramme de Gant du 24 juin au 18 août . . . . .	50