Méthodes de vision par ordinateur et d'apprentissage profond pour la localisation, le suivi et l'analyse de structure de plantes : application au désherbage de précision

Louis Lac

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE SCIENCES POUR L'INGÉNIEUR

AUTOMATIQUE, PRODUCTIQUE, SIGNAL ET IMAGE, INGÉNIERIE

COGNITIQUE

Par **Louis LAC**

Méthodes de vision par ordinateur et d'apprentissage profond pour la localisation, le suivi et l'analyse de la structure de plantes. Application au désherbage de précision.

Sous la direction de : **Jean-Pierre DA COSTA**
Co-encadrant : **Marc DONIAS**

Soutenue le 3 mars 2022

Composition du jury :

M. Jean-Pierre DA COSTA	Professeur	Bordeaux Sciences Agro	Directeur
M. Marc DONIAS	Maître de conférences	Enseirb-Matmeca	Co-encadrant
Mme Laure TOUGNE	Professeure	Université Lumière Lyon	Rapporteure
M. Christophe DEBAIN	Chargé de recherches	INRAE Clermont-Ferrand	Rapporteur
M. Olivier LY	Professeur	Université de Bordeaux	Président
Mme Marine LOUARGANT	Docteure	CTIFL Lanxade	Examinateure

Remerciements

Tout d'abord, je tiens à remercier les rapporteurs d'avoir accepté d'analyser et de rapporter mon travail de thèse ainsi que les membres du jury d'avoir assisté à la soutenance.

Je remercie également mon directeur de thèse, Jean-Pierre Da Costa et mon co-encadrant Marc Donias de m'avoir donné l'opportunité de contribuer à ce beau projet scientifique et de m'avoir accompagné pendant ces trois ans.

Je remercie vivement les membres du projet BIPBIP dont Barna, Hugo et Steve pour les nombreux échanges, notamment au cours des expéditions de terrain, qui ont été des expériences enrichissantes et stimulantes !

J'adresse également mes remerciements aux membres du laboratoire IMS et aux doctorants que j'ai pu côtoyer tout au long de ma thèse : Florent, Ahmed, Marwane, Aymeric, Julien et beaucoup d'autres, pour leur aide précieuse et leur accueil chaleureux.

Enfin, un grand merci à celles et ceux qui ont permis de perfectionner mon manuscrit par leur (long) travail de relecture ainsi qu'aux membres de ma famille et amis qui m'ont soutenu lors de cette aventure.

Table des matières

Table des figures

Liste des tableaux

Introduction

L'agriculture fait aujourd'hui face à des défis de taille. La demande alimentaire est en constante augmentation étant donné la croissance démographique soutenue à l'œuvre depuis 1950 [14]. À ceci s'ajoute la crise environnementale dont l'agriculture est en partie responsable et qui risque d'impacter sa capacité de production dans un futur proche [61]. Limiter ses impacts négatifs tout en assurant des rendements de production permettant d'assurer la sécurité alimentaire mondiale est donc une nécessité.

L'agriculture de précision est l'un des outils participant à la transition vers une agriculture plus résiliente et moins polluante. Celle-ci vise à développer des méthodes de traitement agricole agissant à une granularité spatiale et temporelle plus fine que les méthodes traditionnellement utilisées en agriculture intensive. La décision de l'action est guidée par une analyse plus approfondie des cultures et des besoins [141, 108, 145]. Ces méthodes permettent une action plus localisée et déclenchée au moment le plus opportun afin de minimiser l'impact environnemental des outils et produits employés [42, 71, 46]. L'agriculture de précision s'appuie donc sur un mode de travail plus raisonné et plus spécifique que les méthodes intensives traitant les cultures de manière homogène. Par exemple, l'agriculture de précision s'oppose au traitement chimique d'une culture dans sa totalité. Elle favorisera un traitement local des zones fortement infestées, ou préconisera une irrigation focalisée sur les plantes en état de stress hydrique plutôt que sur l'ensemble de la parcelle.

L'agriculture de précision requiert des moyens d'analyse afin de détecter les besoins, et des moyens d'action pour traiter les zones concernées. Ils peuvent tout à fait être fournis par l'humain au travers de son expertise agricole et de son travail, éventuellement assisté d'outils. Néanmoins, le numérique et les évolutions récentes des matériels de calcul constituent une opportunité d'automatisation des tâches d'agriculture de précision [101]. Plus spécifiquement, les avancées réalisées dans le domaine de la vision par ordinateur[1] accélèrent la robotisation de nombreuses tâches dans le domaine agricole. L'apprentissage profond, un sous-domaine de l'intelligence artificielle qui exploite l'apprentissage automatique via les réseaux de neurones, regroupe des méthodes algorithmiques démontrant un fort potentiel en vision par ordinateur [6, 74] qui commencent à être exploitées pour l'agriculture de précision. Ainsi, de nombreux domaines font l'objet de travaux de recherche et

[1] Les algorithmes en charge de l'analyse et de l'interprétation de l'environnement par la machine.

de développement dans cette direction comme l'estimation du rendement [10], la détection de maladie [139] ou encore le désherbage de précision [94].

Ce manuscrit explore l'utilisation d'algorithmes de vision par ordinateur et d'apprentissage profond pour des applications en désherbage de précision et en surveillance de l'état des cultures. Il s'inscrit dans le cadre du projet BIPBIP qui vise à concevoir un bloc-outil de désherbage mécanique de précision susceptible d'être embarqué sur un porteur agricole ou un robot autonome. Ce bloc-outil capte des données par le biais d'une caméra couleur et procède au désherbage des cultures via des outils de binage mécanique. Il nécessite des algorithmes d'analyse d'images permettant d'obtenir des informations très localisées sur les cultures à traiter afin d'opérer au plus proche de celles-ci. Ce travail propose plusieurs approches de vision par ordinateur et d'apprentissage profond permettant d'une part de détecter et de caractériser des plantes dans des images et d'autre part de suivre certaines parties des plantes dans des séquences d'images.

Le premier chapitre présente le projet BIPBIP dans lequel le travail proposé s'inscrit. Celui-ci dresse ensuite un état de l'art sur les méthodes de vision par ordinateur et d'apprentissage profond employées en agriculture de précision. Finalement, les motivations du travail présenté et les contributions proposées sont détaillées.

Le deuxième chapitre détaille les contributions algorithmiques proposées. Il étudie deux méthodes d'apprentissage profond pour localiser les organes de plantes et estimer le nombre de feuilles : la première est basée sur un détecteur d'objets et la seconde sur une architecture spécifique à la détection de structure de plantes. Ce chapitre présente ensuite une méthode d'agrégation de points d'intérêt pour réaliser du suivi de plantes dans des séquences d'images.

Le troisième chapitre détaille les expérimentations menées sur les hyperparamètres des algorithmes afin de déduire leur valeur optimales. Ce chapitre analyse et compare les performances des algorithmes.

Chapitre 1

Méthodes de vision par ordinateur et apprentissage profond pour l'agriculture de précision : contexte et motivations pour le désherbage de précision

Sommaire

Le travail présenté dans ce mémoire explore l'utilisation d'algorithmes de vision par ordinateur et d'apprentissage profond pour des applications en désherbage de précision et en surveillance de l'état des cultures en plein champ. Il s'appuie sur un prototype de bloc-outil de désherbage mécanique de précision nommé BIPBIP. La section 1.1 présente le projet BIPBIP et les implications sur les besoins des algorithmes développés.

La section 1.2 détaille ensuite un état de l'art sur les méthodes de vision par ordinateur employées en agriculture numérique. Notamment, deux ensembles de méthodes d'intelligence artificielle utilisées pour la classification, la détection et la segmentation de données sont présentés : les méthodes d'apprentissage automatique et les méthodes d'apprentissage profond. L'utilisation de ces méthodes dans des applications de robotique agricole telles que le désherbage de précision et la détection de maladies est également documentée.

Enfin, la section 1.3 présente les motivations du travail de ce mémoire puis introduit les contributions réalisées.

FIGURE 1.1 : Illustration du troisième prototype du bloc-outil BIPBIP embarqué sous un porteur électrique E-Tract conçu par ELATEC.

1.1 Cadre applicatif du désherbage de précision : projet BIPBIP et Challenge ROSE

Le projet BIPBIP[1] (Bloc-outil et Imagerie de Précision pour le Binage Intra-rang Précoce) est un projet de développement d'une solution innovante de désherbage de précision de l'intra-rang en cultures maraichères et grandes cultures à un stade précoce. Cette solution repose sur un bloc-outil de binage équipé d'un système de vision et d'un système décisionnel contrôlant un dispositif de binage entièrement mécanique. Ce bloc-outil est destiné à être embarqué sur un porteur robotisé ou non et est conçu pour être aussi indépendant que possible. Le troisième prototype est illustré dans la figure 1.1. Il est embarqué sous le porteur électrique E-Tract conçu par ELATEC.

Le projet de recherche BIPBIP est financé par l'Agence Nationale de la Recherche (ANR) au travers du dispositif Challenge ROSE[2] (Robotique et Capteurs au Service d'Ecophyto) qui a pour objectif d'encourager le développement de solutions de désherbage intra-rang utilisant peu ou pas d'herbicides. Quatre projets, dont BIPBIP, sont mis en compétition au travers de ce challenge. Les solutions développées sont évaluées par le Laboratoire National de Métrologie et d'Essais (LNE) et l'Institut National de Recherche pour l'Agriculture, l'Alimentation et l'Environnement (INRAE) regroupés au sein du consortium Operose chargé de l'organisation des campagnes d'évaluation menées plusieurs fois par an.

Le bloc-outil est développé en partenariat avec plusieurs acteurs. L'entreprise ELATEC participe à la conception et à la fabrication du bloc-outil et du porteur électrique. Le LaBRI (Laboratoire Bordelais de Recherche en Informatique) est chargé de la robotisation et de

[1] `http://challenge-rose.fr/projet/bipbip/`
[2] `http://challenge-rose.fr`

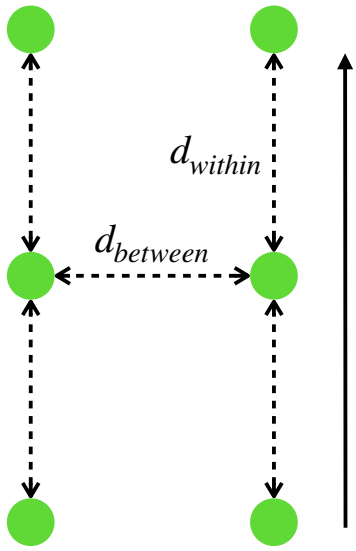


FIGURE 1.2 : Schéma de l'espacement entre les cultures. La flèche indique le sens d'avancement du bloc-outil, $d_{widthin}$ est la distance entre deux plants successifs (points verts) et $d_{between}$ est la largeur entre deux rangs de culture.

l'intégration des algorithmes de vision dans le système décisionnel du bloc-outil. L'équipe MOTIVE du laboratoire IMS (Intégration du Matériau au Système) conçoit le système et les algorithmes de vision intégrés au bloc-outil. Enfin, le CTIFL (Centre Technique Interprofessionnel des Fruits et Légumes) et les Fermes Larrère fournissent les parcelles expérimentales et l'expertise agronomique nécessaires à l'acquisition et l'annotation de données et aux tests des algorithmes et du système mécanique de désherbage. Les travaux de thèse soutenus dans ce mémoire vise à développer des algorithmes de discrimination de culture et d'adventices[3] destinés à être intégrés dans le bloc-outil de désherbage mécanique BIPBIP et prend donc part aux efforts de développements logiciels portés par l'équipe MOTIVE de l'IMS.

Le bloc-outil est conçu pour biner une seule ligne de culture, mais plusieurs modules bloc-outils peuvent être embarqués en parallèle sur le porteur afin de biner plusieurs lignes à la fois. Les cultures visées sont dans un premier temps le maïs et le haricot puisque ce sont celles évaluées par Operose. Néanmoins, des tests ont été réalisés sur le poireau et à terme l'oignon et la carotte sont envisagés. La période de désherbage visée se situe au stade précoce de développement des cultures, c'est-à-dire entre deux et cinq semaines après le semis ou le repiquage. Cette période est la plus critique pour le bon développement des plantes car la compétition avec les adventices y est forte, rendant indispensable une ou plusieurs actions de désherbage, non seulement entre les rangs de cultures mais aussi sur le rang lui-même, entre les plants de culture. L'espacement entre les lignes de culture et les plantes sur le rang est illustré dans la figure 1.2. La largeur moyenne de l'inter-rang $d_{between}$ (distance entre les lignes de cultures) se situe entre 75 cm et 80 cm pour le mais

[3]Mauvaises herbes.

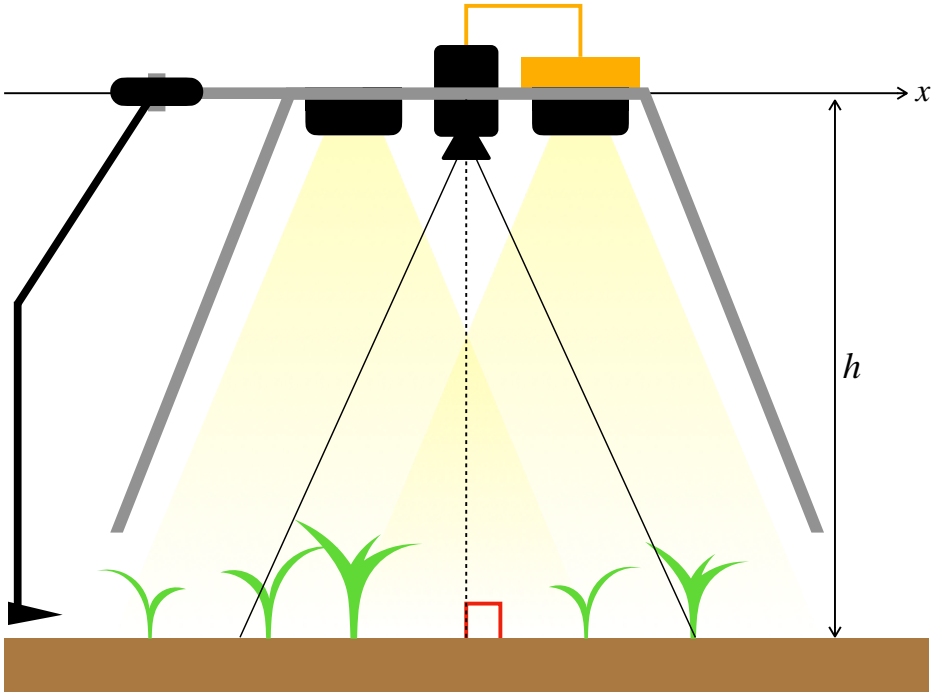


FIGURE 1.3 : Coupe schématique sagittale du module BIPBIP. L'outil de binage mécanique de l'intra-rang est représenté par la tige à gauche, le système décisionnel en jaune, les deux panneaux LED et la caméra en noir à l'intérieur de la chambre de vision (en gris) qui permet d'isoler le système de vision des conditions lumineuses changeantes. h est la hauteur du module de vision par rapport au sol.

et entre 15 cm et 37 cm pour le haricot. La distance moyenne intra-rang d_{within} (distance entre les cultures dans un même rang) est de 15 cm pour le mais et entre 3 cm et 8 cm pour le haricot. De par sa conception décrite plus loin, le bloc-outil est capable de biner tous types d'adventices dans l'inter-rang et l'intra-rang.

Le bloc-outil est conçu autour de trois composants illustrés par la figure 1.3 :

- Un système de vision chargé d'acquérir des images des cultures. Il est composé d'une caméra RGB industrielle Basler acA2500-gc de 3 millions de pixels équipée d'un objectif Basler C125-0418-5M F1.8 f 4 mm. Une chambre de vision (en gris sur la figure 1.3) isole la caméra de la lumière extérieure et deux panneaux LED de 20 W fournissent une illumination contrôlée et homogène afin que les conditions lumineuses soient les plus constantes possibles. La caméra est disposée face au sol (l'axe optique est perpendiculaire au plan du sol) et sa hauteur h (distance du capteur optique au sol) est fixe. Sur le dernier prototype, la hauteur h est de 45 cm. La fréquence d'acquisition est fixée à 10 images par seconde. La figure 1.4 illustre deux images acquises sur le haricot (à gauche) et sur le maïs (à droite) par le système de vision.

- Un système de calcul embarqué chargé de traiter les données en temps réel. Il est constitué d'un ordinateur NVIDIA Jetson Xavier spécialisé pour l'exécution d'algorithmes de vision par ordinateur et d'apprentissage profond. Il a une puissance

(a) Haricot (b) Maïs

FIGURE 1.4 : Illustration d'images acquises par le système de vision de BIPBIP sur une culture de haricot (à gauche) et de maïs (à droite). L'outil de binage composé de deux socs est visible à gauche des images et la jupe de protection de la chambre de vision est visible en haut et en bas des images.

maximale de 30 W.

- L'outil de binage est composé d'actionneurs mécaniques et d'un ou plusieurs outils de travail du sol. La dernière version du prototype en date utilise deux socs latéraux qui s'ouvrent et se referment afin d'éviter les cultures et de détruire les adventices. Le prototype précédent utilisait également un doigt mécanique grattant le sol et capable de se déporter sur le côté de la ligne de culture pour éviter de détruire les plantes d'intérêt. Le fonctionnement de cet outil n'est pas étudié plus en détail dans ce manuscrit, seuls ses prérequis de fonctionnement sont nécessaires et conditionnent la conception des algorithmes de discrimination de plantes et d'adventices. Plus spécifiquement, l'outil de binage requiert la position précise du point d'entrée de la tige de la plante dans le sol afin d'éviter de la déraciner.

Il n'y a pas de système de positionnement absolu (GPS, RTK) ni de système de positionnement relatif (centrale inertielle) intégrés dans le bloc-outil BIPBIP afin de diminuer le coût et de rendre l'outil plus autonome (il ne dépend pas de la bonne réception du signal GPS notamment).

Outre l'intérêt agronomique d'un binage superficiel, l'avantage de ce fonctionnement est qu'il n'y a pas besoin de détecter, de classifier ou de localiser les adventices, seules les positions des cultures, ou plus précisément, les positions des points d'entrée de leurs tiges dans le sol, sont nécessaires. Cela réduit d'autant le temps d'annotation et permet au système de ne pas être spécifique à un nombre de types d'adventice prédéfini.

Certaines spécificités du système BIPBIP ont orienté les choix algorithmiques présentés dans ce travail, notamment la nature des images collectées (image en champ rapproché, vue de dessus, luminosité contrôlée, etc.) et les contraintes de déplacement et d'orientation

de la caméra (mouvement linéaire, plan de la caméra parallèle au sol, etc.). La nature de ces choix et les hypothèses réalisées sont détaillées dans la section 2.1.

1.2 Méthodes de vision par ordinateur et d'apprentissage profond pour l'agriculture de précision

La vision par ordinateur est un levier permettant d'automatiser certaines tâches d'agriculture de précision, notamment de désherbage de précision. C'est un composant essentiel des applications en robotique agricole et en proxidétection (détection sans contact mais réalisée au plus près des cultures) [120].

La vision par ordinateur est une brique algorithmique qui s'intègre soit dans un système d'aide à la décision ou d'expertise, soit dans une plateforme robotique. Cette intégration est résumée dans la figure 1.5. Les algorithmes de vision par ordinateur s'appuient sur des données issues de capteurs (en bleu dans la figure 1.5), par exemple des images en couleur, des images de profondeur ou des données GPS. Ces données sont ensuite analysées par les algorithmes de vision par ordinateur (en vert sur la figure 1.5) afin d'en créer une représentation plus abstraite et de plus haut niveau. Cette information peut ensuite être exploitée de deux manières :

- L'information est directement présentée à un opérateur humain afin de l'assister pour une expertise ou une décision. Cela peut être un indicateur d'infestation d'une culture par une maladie, par exemple. L'information peut aussi être archivée pour une future utilisation.

- L'information est exploitée par d'autres algorithmes afin de commander un système automatisé (en jaune sur la figure 1.5). Par exemple, l'information peut servir à contrôler un système d'irrigation parcimonieuse ou de pulvérisation de précision.

Dans le cas de BIPBIP le capteur est une caméra couleur et les actionneurs sont des moteurs et des outils de désherbage de précision de l'intra-rang.

Le travail proposé dans ce manuscrit se focalise sur des méthodes dites "d'apprentissage automatique" (*Machine Learning* (ML)). Ces méthodes visent à automatiser l'apprentissage et la réalisation d'une tâche par une machine (un ordinateur). Dans le cadre de la vision par ordinateur, une liste des tâches les plus courantes est la suivante :

- La classification attribue une donnée à une catégorie. Par exemple, la classification d'une image de vigne comme malade ou non malade.

- La régression assigne une ou plusieurs valeur(s) numérique(s) à une donnée. Par exemple, un indice de développement d'une culture basé sur une échelle de stades phénologiques ou le besoin en irrigation d'une culture.

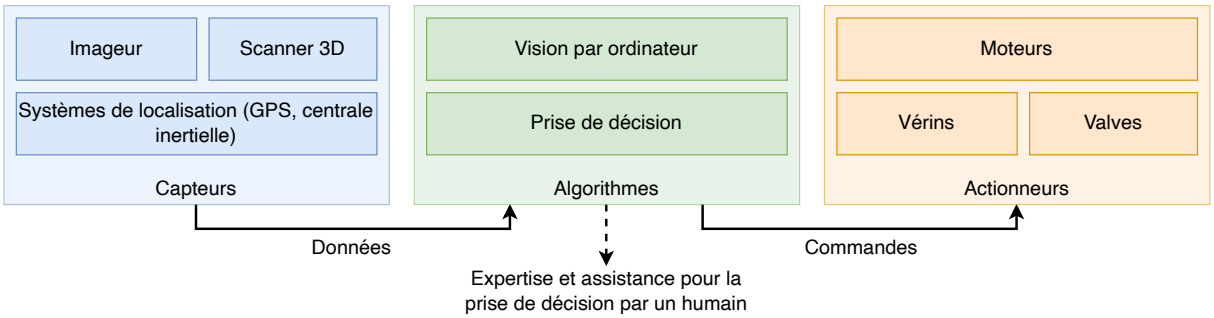


FIGURE 1.5 : Schéma synthétique des principaux composants d'une application de la vision par ordinateur et du numérique pour l'agriculture de précision. L'automatisation des tâches repose sur l'extraction de données issues de capteurs (en bleu), traités et analysés par des algorithmes (en vert) et permettant de commander un automate (en jaune) ou d'assister un opérateur humain.

- La détection correspond à la localisation de plusieurs objets dans la donnée. Par exemple, les boîtes englobantes de baies sur un arbre ou les boîtes englobantes de plantes dans un rang de culture.
- La segmentation est une classification plus fine. Les algorithmes de segmentation donnent une information de classification par zone ou par pixel. Par exemple, la discrimination entre les cultures et les adventices au niveau du pixel rentre dans cette catégorie.

Les méthodes d'apprentissage automatique sont généralement classée comme un sous-domaine des méthodes d'intelligence artificielle (*Artificial Intelligence* (IA)) [79, 19] mais cette affirmation fait débat dans la communauté, certains classant l'IA comme un domaine orthogonal au ML[4] [12, 7, 99]. Un ensemble de méthodes dites "d'apprentissage profond" (*Deep Learning* (DL)), basée sur des réseaux de neurones artificiels (*Artificial Neural Networks* (ANN)) et étant comprises dans le domaine des approches d'apprentissage automatique vont plus particulièrement nous intéresser. La figure 1.6 schématise la hiérarchie de ces différents domaines la plus couramment admise dans la littérature.

Afin de présenter les avantages et inconvénients des méthodes d'apprentissage profond et de les comparer plus simplement aux autres méthodes d'apprentissage automatique, deux ensembles d'approches de ML sont identifiées et décrites dans les sections suivantes :

- Les approches que nous qualifierons "classiques" d'apprentissage automatique. Cette catégorie regroupe notamment des méthodes d'extraction de descripteurs et des algorithmes décisionnels. Ces approches et leur emploi en agriculture de précision sont détaillés dans la sous-section 1.2.2.
- Les méthodes d'apprentissage profond. Bien que faisant partie des méthodes d'apprentissage automatique, celles-ci sont regroupées à part afin de mieux distinguer

[4]Le degré d'intelligence artificielle est difficile à caractériser, certains considèrent qu'un algorithme de ML n'est pas forcément "intelligent".

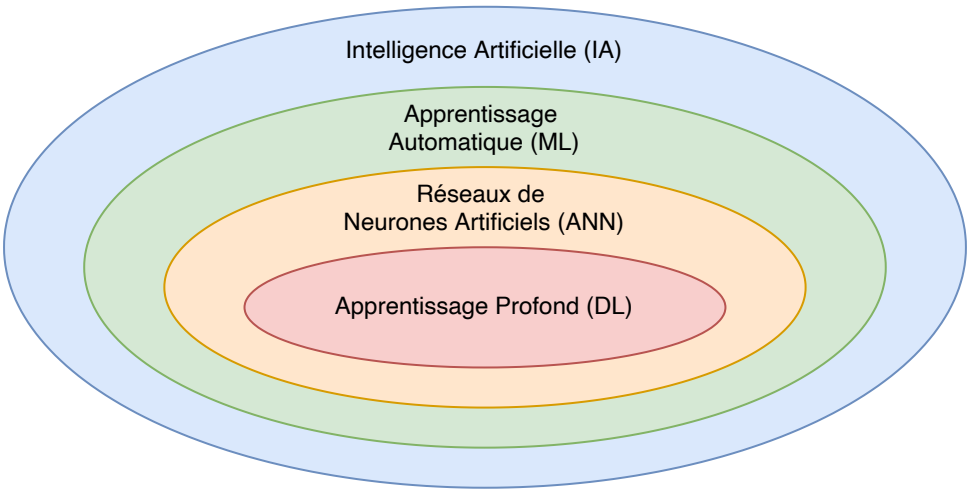


FIGURE 1.6 : Schéma synthétique de la hiérarchie entre les principaux des domaines de l'intelligence artificielle.

leurs particularités. Ces méthodes et leurs applications en agriculture de précision sont détaillées dans la section 1.2.3. En raison du lien de parenté avec les méthodes d'apprentissage automatique, certains aspects sont couverts dans la sous-section 1.2.2.

Additionnellement, la sous-section 1.2.1 détaille les principaux capteurs de données employés pour générer les données utiles aux méthodes d'apprentissage automatique.

1.2.1 Capteurs de données

Une grande variété de capteurs est employée dans les systèmes de vision par ordinateur utilisés en robotique agricole, notamment pour des applications en désherbage de précision. Cette section présente rapidement les capteurs plus utilisés ainsi que leurs applications respectives.

Les caméras couleur (caméras RGB) sont les capteurs les plus employés car ils sont bons marchés et bénéficient d'une large bibliothèque d'algorithmes de vision par ordinateur existants. Ils peuvent être utilisés pour classifier des espèces de plante [34], segmenter des adventices [9], discriminer des plantes d'intérêt et des adventices [40] ou encore pour détecter des rangs de cultures [100].

Les caméras en proche infrarouge (caméras RGB-NIR) sont également régulièrement employées. Celles-ci sont capables de capter des données sur une longueur d'onde située dans le domaine du proche infrarouge (*Near Infrared* (NIR)) en plus des trois canaux rouge-vert-bleu usuels d'une caméra couleur. Les végétaux reflètent bien ces longueurs d'onde et certains travaux tentent d'exploiter ce phénomène pour améliorer les performances de classification ou de détection de plantes. Ainsi, les caméras RGB-NIR sont utilisées dans certains travaux pour détecter les lignes de cultures afin de réaliser du désherbage autonome [26] ou pour discriminer les plantes d'intérêt des adventices [53].

Plus rarement, des imageurs employant davantage de longueurs d'onde sont déployés. C'est le cas des caméras multispectrales (nombre de longueurs d'onde généralement inférieur à dix) et des caméras hyperspectrales (entre une centaine et un millier de longueurs d'onde). Ces capteurs sont globalement plus onéreux que les caméras RGB et RGB-NIR et comportent des difficultés techniques rendant leur utilisation plus complexe [49] (caméras plus imposantes, temps de prise de vue long, non-alignement spatial des longueurs d'onde, données gourmandes en mémoire). Néanmoins, ces capteurs fournissent une information spectrale plus riche que les caméras couleur, ce qui peut permettre de mieux détecter certains végétaux. Certains travaux emploient par exemple un imageur hyperspectral afin de détecter les adventices pour une application en robotique agricole [134].

Un dernier groupe de capteurs est parfois employé, souvent en parallèle avec d'autres capteurs comme les caméras RGB : les capteurs de distance. On y retrouve les LiDAR (capteurs lumineux de distance), la vision stéréoscopique (impliquant deux caméras RGB pour calculer la profondeur de la scène) et parfois les Sonars (capteurs sonores de distance). Ceux-ci permettent une analyse en trois dimensions plus poussée de la scène observée qui fournit une information supplémentaire pour les tâches de détection et de guidage robotique. Par exemple, un Sonar peut être monté à côté d'une caméra RGB-NIR afin de connaître l'altitude de la caméra pour tracker les cultures et guider un robot de désherbage de précision [136]. Des LiDAR peuvent être employés pour modéliser la scène observée et ainsi assister la navigation automatique d'un robot réalisant des tâches d'agriculture de précision [25]. La profondeur de la scène peut aussi être directement déduite d'un flux vidéo grâce à des algorithmes de reconstruction en trois dimensions et cette information peut être employée pour assister la détection et le phénotypage de plantes [140].

D'autres capteurs indirectement liés à la vision peuvent être employés pour améliorer les performances des autres capteurs : les systèmes de positionnement comme le GNSS (*Global Navigation Satellite System*), la cinématique temps réel (RTK) et les centrales inertielles (IMU).

Dans la pratique, les caméras RGB sont préférées aux caméras multispectrales et hyperspectrales du fait de leur coût limité, de leur versatilité et de leur simplicité d'emploi. Les capteurs de distance sont souvent intégrés mais servent principalement au guidage et à la navigation de robots. Pour ces raisons le bloc-outil BIPBIP intègre une unique caméra RGB pour la localisation de plantes d'intérêts et d'organes de plantes[5]. Les méthodes de vision par ordinateur évoquées dans les deux sections suivantes sont principalement axées sur les algorithmes de traitement d'images en couleur.

[5]Un LiDAR est présent mais n'est utilisé que pour la décision d'action mécanique.

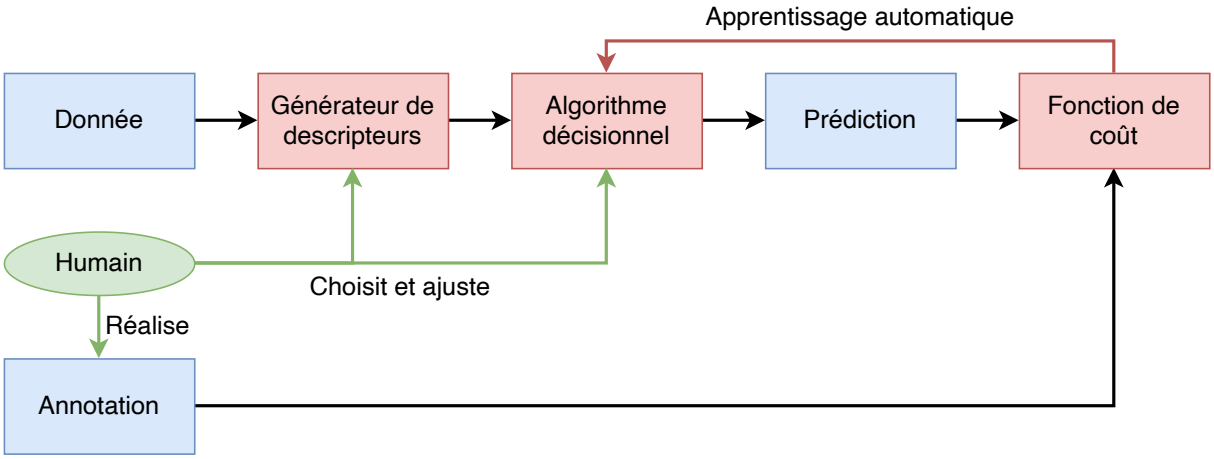


FIGURE 1.7 : Représentation schématique d'un algorithme d'apprentissage automatique supervisé. Un expert humain choisit le générateur de descripteur le plus pertinent basé sur ses connaissances spécifiques du domaine d'application. L'algorithme décisionnel est entraîné automatiquement via les données d'apprentissage.

1.2.2 Méthodes classiques d'apprentissage automatique pour la vision par ordinateur et applications en agriculture de précision

Les algorithmes d'apprentissage automatique sont basés sur des méthodes visant à automatiser l'apprentissage d'une tâche par la machine, par exemple la classification d'images et la détection d'objets dans des vidéos. L'ensemble des méthodes d'apprentissage automatique peut à nouveau être divisé en plusieurs branches selon la quantité et la qualité des données requises pour l'apprentissage :

- Les approches supervisées qui nécessitent des données d'apprentissage annotées, i.e un annotateur humain doit fournir des exemples de résultats attendus pour que l'algorithme apprenne la tâche demandée.
- Les approches non supervisées pour lesquelles l'algorithme apprend seul la structure sous-jacente des données sans annotation.
- Les approches semi-supervisées (*Self-Supervised*) qui sont un mélange des deux approches précédentes.

Les méthodes de vision par ordinateur les plus courantes et qui sont décrites dans la suite utilisent principalement l'approche supervisée. L'apprentissage profond décrit plus loin se distingue des méthodes d'apprentissage automatique "classiques" par une intervention humaine moindre : l'apprentissage est entièrement automatique. La suite de la section se focalise sur les méthodes d'apprentissage automatique classiques mais les remarques sont également valables pour les approches profondes dans une certaine mesure.

Les algorithmes de ML supervisés sont composés de deux briques algorithmiques mises en série : (i) un algorithme de génération de descripteurs (*features* dans la littérature anglophone) qui extrait des indicateurs pertinents pour décrire le problème et analyser la donnée, par exemple pour une image cela peut correspondre à des textures, des formes ou des indices colorimétriques qui sont propres aux objets que l'on voudrait détecter dans ces images, et (ii) un algorithme décisionnel qui analyse les descripteurs afin de générer une proposition, i.e. une information de haut niveau qui répond à la tâche demandée, par exemple une liste de détections d'objets dans une image. La figure 1.7 illustre le fonctionnement de l'apprentissage : la donnée est d'abord analysée pour générer des descripteurs puis ceux-ci sont traitées afin d'en déduire la prédiction. Un expert humain sélectionne le générateur de descripteurs qu'il juge le plus pertinent vis-à-vis de ses connaissances spécifiques du domaine d'application. L'algorithme décisionnel est ensuite entraîné automatiquement via une métrique évaluant la performance de la prédiction sur les données d'entraînement (données et annotations).

Voici un exemple pour une application en classification d'espèces d'arbres basée sur des images de feuilles. Les connaissances spécifiques de l'expert du domaine lui permettent de constater que chaque essence d'arbre à classifier a une forme de feuille différente des autres espèces. L'expert décide donc d'utiliser des méthodes d'extraction de descripteurs spécifiques à la forme des feuilles. Il sélectionne ensuite un algorithme décisionnel de classification (éventuellement capable d'être appris automatiquement) afin de catégoriser l'espèce des arbres.

Les deux sections suivantes détaillent ces deux briques : l'extraction de descripteurs et les algorithmes de prise de décision.

1.2.2.1 Méthodes d'extraction manuelle de descripteurs

Les méthodes d'extraction de descripteurs se réfèrent à un ensemble de techniques d'analyse géométrique, fréquentielle et statistique du signal et de l'image permettant de construire une représentation de plus haut niveau d'abstraction des données brutes. Elles permettent de décrire la donnée plus finement ou de mettre en avant certains aspects ayant un intérêt particulier pour la décision [120]. Ces méthodes d'extraction de descripteurs sont basées sur l'ingénierie des descripteurs (*feature engineering*), c'est-à-dire la recherche empirique des meilleurs algorithmes d'extraction de descripteurs, basée sur l'intuition, l'observation des données et les connaissances spécifiques du domaine d'application. Par exemple, concevoir un filtre de convolution basé sur la présence d'une texture récurrente dans une image ou opérer une transformation colorimétrique qui favorise une teinte caractéristique de l'objet à détecter.

Les algorithmes d'extraction manuelle de descripteurs sont basés sur les outils d'analyse statistique et fréquentielle du signal et de l'image. Cela inclut, de manière non exhaustive, les algorithmes de détection de contours (Sobel, Prewitt, Canny, Laplacien, etc.), les

détecteurs de points d'intérêt (filtre de Harris [51], SIFT [95], transformée de Hough linéaire ou circulaire [52]), la caractérisation de texture (transformée en Ondelettes [29], transformée de Fourier et le Tenseur Local de Structure [72]) et les méthodes de traitement d'image classiques telles que les opérateurs morphologiques, les opérations colorimétriques (changement d'espace de couleur, Excess Green Index (EGI) [135], Normalized Difference Vegetation Index (NDVI) [23], et autres indices de végétation).

Un avantage de ces méthodes-là est qu'elles ne nécessitent pas d'apprentissage sur une base de données au préalable : elles peuvent être employées directement, moyennant quelques ajustements d'hyper-paramètres. En revanche, elles souffrent d'une faible capacité de généralisation (elles donnent de mauvais résultats sur un jeu de données qui s'éloigne des conditions observées initialement, phénomène plus connu sous le nom de *domain shift* dans la littérature) et sont souvent complexes à développer [68]. Par exemple l'EGI n'est pas robuste aux changements de couleur et échoue sur de la végétation qui n'est pas majoritairement verte, les détecteurs de contours fonctionnent mal sur les objets peu texturés ou aux contours mal définis. Le choix des descripteurs est donc dépendant du type de données auxquelles on s'intéresse et est laissé à la discrétion du praticien ou plutôt de l'expert en charge du développement de l'algorithme de vision par ordinateur.

La détection d'adventices et de maladie ainsi que le désherbage de précision font partie des applications qui utilisent ces techniques d'extraction de descripteurs. Par exemple, le NDVI et la transformée de Hough linéaire peuvent être appliqués à des images hyperspectrales afin d'obtenir un descripteur de végétation pouvant ensuite être utilisé pour discriminer les rangs de culture entre eux [134]. Les lignes de cultures peuvent aussi être détectée sur des images en couleur par le biais de l'EGI [100]. Une autre application possible est la discrimination des plantes et des adventices basée sur la transformée en ondelettes et l'EGI [15]. Celle-ci peut aussi être réalisée avec l'indicateur NDVI appliqués à des images en proche infrarouge [54]. Finalement, d'autres travaux en détection de maladies de la vigne emploient le tenseur de structure [1] afin de détecter le mildiou.

1.2.2.2 Algorithmes décisionnels

Les algorithmes décisionnels désignent les algorithmes qui opèrent sur un ensemble de descripteurs pour en tirer une information de haut niveau. En vision par ordinateur, cette information de haut niveau correspond souvent à une classification, une segmentation, une détection d'objets ou encore une régression (un niveau d'infestation par exemple ou un concept plus complexe comme des lignes de cultures).

On retrouve dans cette catégorie des approches paramétriques comme l'analyse discriminante linéaire (*Linear Discriminant Analysis* (LDA)) [63] ou plus généralement la classification Bayésienne ainsi que des approches non paramétriques telles que la classification des K plus proches voisins (*K-Nearest Neighbors* (KNN)) [38], les Machines à Vecteurs de Support (SVM) [129] utilisées pour la classification et la régression, et les

arbres de décision ou forêts aléatoires (*Random Forest*) [18] utilisés pour la classification. On retrouve aussi dans cette catégorie les réseaux de neurones artificiels (ANN) [96] peu profonds (avec peu de neurones et de couches cachées). Ces méthodes sont principalement supervisées, elles ne requièrent pas ou peu d'apprentissage et sont souvent rapides à mettre en œuvre. Elles sont en revanche peu robustes car elles nécessitent la définition manuelle d'hyperparamètres qui dépendent de l'application visée. Par exemple, le KNN requiert le choix d'une valeur du nombre de voisins K votant pour la décision finale et les forêts aléatoires nécessitent de fixer des paramètres comme le nombre d'arbres de décision. Le SVM est, par certains aspects, similaire à un réseau de neurones [27] mais dans la pratique il est souvent jugé moins performant lorsque la base de données d'entraînement est conséquente et la dimension des données élevée [27, 35]. La plupart de ces algorithmes, notamment le SVM et LDA, sont sensibles au fléau de la dimension (*Curse of Dimensionality* dans la littérature) [35] et il est difficile de trouver des règles de décision robustes.

En robotique agricole et agriculture numérique, ces méthodes sont largement utilisées pour segmenter et classifier la végétation ou les maladies. Ainsi, la discrimination entre les pixels de plantes d'intérêt et ceux d'adventices dans des images RGB est souvent réalisée avec le LDA ou le SVM [134]. Les forêts aléatoires peuvent aussi être utilisées pour segmenter les plantes et ainsi les discriminer des adventices [53]. L'estimation de la position des lignes de culture est quant à elle souvent réalisée par une régression linéaire via les moindres carrés appliquée à un masque de la végétation obtenu par seuillage d'Otsu [104] d'un indicateur colorimétrique calculée sur des images RGB ou NIR [100]. Finalement, la classification de parties de plantes comme les organes de la vigne est parfois réalisée via une classification Bayésienne [2].

La section suivante illustre les particularités de l'apprentissage profond par rapport aux méthodes d'apprentissage automatique plus traditionnelles précédemment présentées et détaille les usages en agriculture de précision.

1.2.3 Méthodes d'apprentissage profond pour la vision par ordinateur et applications en agriculture de précision

L'apprentissage profond est une branche de l'apprentissage automatique décrit plus haut. Les considérations sur la composition des algorithmes qui en font partie (la génération de descripteurs et les algorithmes décisionnels) et sur les différentes approches possibles (approches supervisées et non supervisées) sont aussi valables ici. Les algorithmes d'apprentissage profond sont basés sur les réseaux de neurones artificiels (*Artificial Neural Networks* (ANN)) qui sont chargés d'apprendre automatiquement les deux briques du ML : la génération de descripteurs et la prise de décision. Les principales différences avec le ML sont que les algorithmes de DL ont un besoin plus faible en intervention humaine et qu'ils sont basés sur des réseaux de neurones dits "profonds" (*Deep Neural Networks* (DNN)),

c'est-à-dire avec un nombre de neurones élevé.

1.2.3.1 Principes de fonctionnement des algorithmes d'apprentissage profond

Cette sous-section résume de manière synthétique le fonctionnement des réseaux de neurones artificiels profonds. Un état de l'art plus détaillé et développant la théorie des réseaux de neurones artificiels (ANN) est présenté dans l'annexe A.

D'un point de vue formel, un réseau de neurones artificiel désigne une fonction paramétrique différentiable opérant sur un ensemble $\mathbb{R}^n,\ n \in \mathbb{N}$ et produisant des valeurs dans un ensemble $\mathbb{R}^m,\ m \in \mathbb{N}$:

$$y = \mathcal{F}_\theta(x),\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m. \tag{1.1}$$

Il est théoriquement possible de faire "apprendre" n'importe quelle fonction $\mathcal{F}$ de $\mathbb{R}^n$ vers $\mathbb{R}^m$ au réseau de neurones, c'est-à-dire de régler les paramètres θ du réseau de manière à approcher la fonction $\mathcal{F}$ [58]. Ce réglage des paramètres θ du réseau peut être réalisé efficacement par un algorithme de descente de gradient [16] basé sur les méthodes de rétro-propagation [3]. Pour fonctionner, ces méthodes requièrent (dans le cas d'un apprentissage supervisé) des échantillons $\hat{x} \in \mathbb{R}^{n\times k},\ k \in \mathbb{N}$ ainsi que des exemples de résultat attendus $\hat{y} \in \mathbb{R}^{m\times k},\ k \in \mathbb{N}$. Autrement dit, en ne disposant que d'observations partielles de la fonction inconnue $\mathcal{F}$, on peut faire apprendre à un réseau une approximation $\mathcal{F}_\theta$ de $\mathcal{F}$.

D'un point de vue applicatif, la fonction $\mathcal{F}$ peut modéliser une tâche potentiellement complexe comme de la détection d'objets dans des images. Dans ce cas, la donnée d'entrée $x \in \mathbb{R}^{H\times W\times C}$ serait une image couleur de taille $H \times W$ et la donnée de sortie $y \in \mathcal{R}^{K\times 4}$ serait un ensemble de K boîtes englobantes ayant chacune 4 coordonnées spatiales exprimées dans le système de coordonnées de l'image.

Les réseaux de neurones convolutifs font partie d'un sous-ensemble des réseaux de neurones artificiels spécialisés dans le traitement de données comportant des cohérences locales (spatiale ou temporelle) telles que les images. Ils sont basés sur le partage des poids synaptiques (les paramètres des neurones) permettant d'implémenter efficacement certaines opérations du réseau de neurones sous la forme de convolutions. L'optimisation apportée par cette méthode permet de réduire radicalement le temps d'entraînement du réseau et d'augmenter sa performance [6].

D'un point de vue architectural, les réseaux convolutifs sont généralement organisés par couches successives. Chaque couche est constituée de filtres de convolution appris par le réseau, de non-linéarités et éventuellement d'opérateurs de normalisation permettant de stabiliser l'entraînement [62]. Les couches sont souvent interprétées comme étant des opérateurs transformant une donnée d'entrée en une donnée de sortie comportant une information de plus haut niveau, plus abstraite. Dans le même temps, chaque couche réduit la "définition" (spatiale ou temporelle) de la donnée d'entrée. Cette réduction poursuit

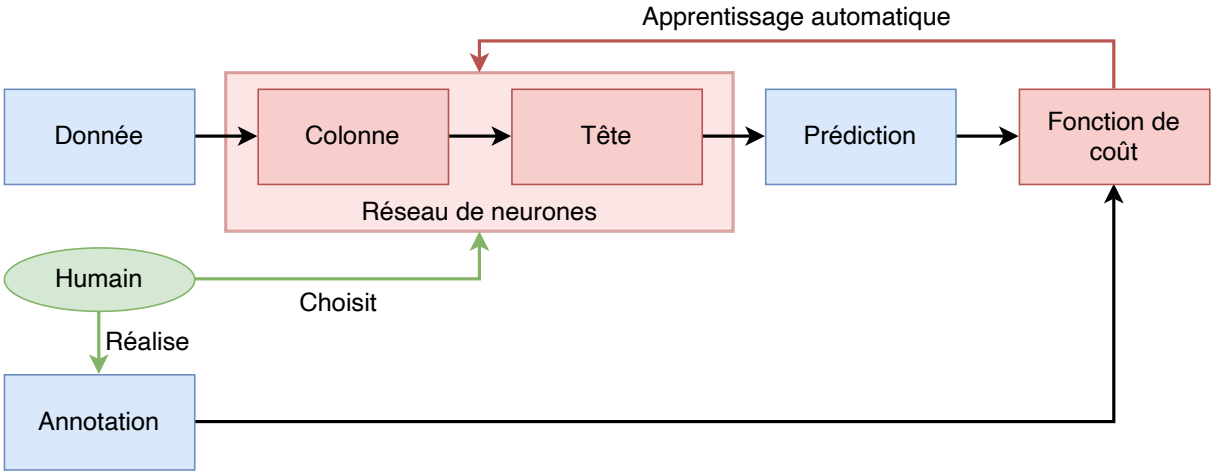


FIGURE 1.8 : Représentation schématique d'un algorithme d'apprentissage profond supervisé. Le réseau de neurones apprend automatiquement la tâche via la supervision par la fonction de coût qui évalue la qualité de la prédiction. Le réseau de neurones est composé de deux parties en série, analogues au générateur de descripteurs et à l'algorithme décisionnel : la colonne et la tête de prédiction (*backbone* et *head* dans la littérature), qui sont appris conjointement.

deux buts : (i) diminuer la charge de calcul du réseau de neurones et (ii) augmenter le champ de vision du réseau, i.e. permettre à la couche suivante d'analyser la donnée à une échelle plus large afin d'améliorer la capacité d'abstraction du réseau.

Un réseau de neurones, convolutif ou non, est dit "profond" lorsqu'il fait intervenir un nombre élevé de neurones ou de couches de neurones. La différence avec les réseaux de neurones "classique" est donc essentiellement une différence d'échelle.

La figure 1.8 illustre le fonctionnement d'un algorithme d'apprentissage profond supervisé. Un réseau de neurones artificiel remplace le générateur de descripteurs et l'algorithme de prise de décision. Il est néanmoins composé de deux composants analogues qui ont les mêmes fonctions : la colonne (*backbone* dans la littérature) extrait des descripteurs et la tête (*head* dans la littérature) prend la décision. Contrairement au ML, les deux composants font l'objet d'un apprentissage et sont généralement appris conjointement. L'apprentissage est supervisé par une fonction de coût qui évalue la qualité de la prédiction du réseau de neurones en fonction des annotations de données.

L'approche supervisée est aujourd'hui dominante dans le domaine de la vision par ordinateur bien que des méthodes moins supervisées commencent à apparaître[6]. Il existe aussi des méthodes hybrides. Par exemple, l'extraction de descripteurs est réalisée avec un algorithme d'apprentissage profond mais l'algorithme de décision est basé sur une méthode d'apprentissage automatique plus classique [5]. Seules les approches entièrement neuronales sont abordées dans ce manuscrit.

[6]Par exemple l'apprentissage par contraste [138] propose de pré-entraîner un générateur de descripteurs de manière non supervisée permettant d'obtenir de meilleures performances lorsqu'il est ensuite entraîné de manière supervisée sur une tâche de détection d'objets.

Les méthodes de traitement de données par apprentissage profond sont de plus en plus populaires depuis les premières démonstrations de leur performance supérieures faces aux approches "classiques" [74]. Elles sont aujourd'hui massivement utilisées dans pratiquement tous les domaines [6, 109] (médecine, prédiction météorologique, conduite autonome, analyse de la parole, traduction automatique, etc.) et y démontrent des performances supérieures autant en précision qu'en temps d'exécution [6, 69, 144], ce qui les rend attractives.

Un autre facteur qui rend ces approches attractives est la meilleure capacité de généralisation et d'abstraction qu'apporte l'aspect entièrement automatique de l'apprentissage profond. Par exemple, un réseau de neurones de détection d'objets n'a pas besoin d'être spécialisé pour des applications différentes, l'architecture du réseau peut être la même pour une application en détection d'humains que pour une application en détection de voitures. Seules les données fournies au réseau de neurones pour son apprentissage varient d'une application à une autre. Les réseaux de neurones sont aussi très flexibles et peuvent apprendre plusieurs tâches simultanément [133]. Il est tout à fait possible d'utiliser le même réseau pour réaliser à la fois de la détection d'objets, de la segmentation et de la classification (en utilisant une colonne partagée et plusieurs têtes) [55].

Néanmoins, les algorithmes d'apprentissage supervisés nécessitent d'énormes quantités de données[7] lors de l'entraînement, ce qui peut être chronophage et coûteux [124]. Aussi, ces méthodes sont souvent qualifiées de boîtes noires (*black boxes*) : les descripteurs et les algorithmes de décision appris étant difficilement interprétables. L'emploi de réseaux de neurones profonds a également l'inconvénient de déporter le besoin d'expertise du domaine de la vision par ordinateur au domaine de l'application visée. En effet, l'annotation des données doit souvent être réalisée par un expert du domaine capable d'identifier les éléments spécifiques (maladies, organes des plantes, espèce des plantes, etc.) dans les données d'entraînement.

1.2.3.2 Applications de l'apprentissage profond en agriculture de précision

L'agriculture numérique a, elle aussi, pris le tournant de l'apprentissage profond [128], mais sa croissance est plus lente et plus récente [91]. C'est notamment le cas dans le domaine du phénotypage de plante où peu de bases de données et peu de développement d'algorithmes de traitement d'image [128] sont disponibles. Cependant, comme dans les autres domaines, l'apprentissage profond remplace progressivement les méthodes de traitement de données traditionnelles et montre des performances supérieures [68]. Les méthodes d'apprentissage profond les plus employées sont principalement des réseaux de neurones d'analyse d'images comme les réseaux de classification d'images [122, 126], les réseaux de détection d'objets [85, 90] et les réseaux de segmentation sémantique d'images [55, 116]. Ces réseaux font

[7] Il existe cependant des méthodes pour réduire le nombre de données à annoter comme la simulation [67, 110, 123] ou l'entraînement semi-supervisé [106].

partie de la classe des Réseaux de Neurones Convolutifs (CNN), leur structure est détaillée en annexe A.

La détection d'adventices et de plantes est aussi un des domaines de l'agriculture de précision qui utilise le plus l'apprentissage profond mais on y trouve également le calcul de rendements agricoles, la détection de maladies et le phénotypage [68]. Les réseaux de neurones de classification sont par exemple employés pour discriminer entre plusieurs dizaines de plantes d'intérêt et d'adventices dans des images RGB [34]. Parfois, les descripteurs d'images sont extraits avec des méthodes plus traditionnelles, comme l'EGI, et le réseau de neurones est uniquement utilisé pour classifier l'espèce des plantes [64]. Quand la classification n'est pas suffisante et qu'il y a besoin d'une information précise sur la localisation des plantes, certains travaux emploient des réseaux de neurones de segmentation d'images pour discriminer les plantes [98] ou pour estimer la composition botanique de cultures [123]. Plus rarement, les réseaux de neurones de segmentation sont appliqués à des séquences d'images RGB, parfois avec un canal NIR, afin de segmenter plus finement la végétation [94]. Ils requièrent cependant une annotation très chronophage puisqu'il faut annoter tous les pixels des images. Dans le domaine du phénotypage, les réseaux de neurones de régression et de détection d'objets sont souvent employés pour estimer le nombre de feuilles des plantes d'intérêt, comme le maïs, dans des images en couleur [97]. Certains travaux estiment d'autres paramètres conjointement au nombre de feuilles tels que la surface foliaire [31]. Ces deux applications sont en revanche limitées à une seule plante par image. Concernant la détection de maladies, certains travaux utilisent des architectures neuronales plus récentes comme que les réseaux de l'attention [130] afin de détecter les feuilles atteintes de maladies [139].

Les réseaux de neurones profonds montrent donc des résultats très prometteurs en matières d'applications agricoles. Moyennant une quantité de données d'entraînement annotées suffisante, ils permettent d'atteindre de bonnes performances dans des tâches variées. Leur bonne capacité de généralisation est un avantage qui autorise leur emploi dans des conditions très changeantes, sans modification de leur fonctionnement. Ainsi, un réseau peut apprendre à détecter plusieurs types de cultures dans des conditions de sol (sol sec, mouillé, sableux, etc.) et de contextes variables (ombres, soleil direct, image "brûlée", etc.). De plus, ces algorithmes bénéficient aujourd'hui d'accélérations matérielles (processeur de tenseurs (TPU), calcul flottant mixte (*Mixed Precision Lerning*), processeur graphique (GPU) embarqué à basse consommation comme le NVIDIA Jetson Xavier, etc.) permettant de les employer sur du matériel embarqué pour une application en temps réel.

Ces considérations nous on fait opter pour l'utilisation d'approches profondes dans les méthodes de vision par ordinateur développées dans ce manuscrit pour une application en désherbage de précision. La section suivante formalise les objectifs et contributions du travail proposé.

1.3 Motivations et contributions

Les sections précédentes ont montré l'intérêt que représente l'apprentissage profond pour des applications relatives à l'agriculture de précision ainsi que les limites et enjeux de ces méthodes. Les hautes performances et la bonne capacité de généralisation de celles-ci ainsi que la possibilité de les employer pour un traitement en temps réel ont motivé le choix de leur utilisation dans les méthodes proposées dans ce manuscrit.

L'objectif de ce travail est de développer des solutions d'apprentissage automatique et de vision par ordinateur permettant de détecter des plantes d'intérêt dans des images, de collecter certaines de leurs propriétés, comme la localisation des organes et le nombre de feuilles, et de suivre leur position dans des séquences d'images afin d'augmenter la précision des détections. Ceci peut profiter à des applications en désherbage de précision qui requièrent la position précise d'organes de plantes et pour des applications en surveillance de l'état des cultures, par exemple pour connaître le stade de développement des cultures. Plus spécifiquement, les outils développés ciblent une application en désherbage mécanique de précision de l'intra-rang, dont le cadre applicatif est détaillé en section 1.1.

Les contributions de ce travail concernent plusieurs maillons de la chaîne de traitements de l'image, de l'acquisition et de l'annotation des données à la conception des systèmes de détection et de leur évaluation. Les contributions majeures sont :

- Le choix, le paramétrage et l'évaluation d'un réseau de référence issu de l'état de l'art des réseaux de détection d'objets pour réaliser des tâches de détection d'organes de plantes et d'estimation du nombre de feuilles. Une méthode d'annotation spécifique est développée et un entraînement non conventionnel du réseau est réalisé afin de localiser des organes de plantes plutôt que des plantes entières et de classifier le nombre de feuilles. La pertinence de cette approche est ensuite justifiée par des expérimentations.

- La conception d'un réseau de neurones profond dédié à la détection de structure de plantes, i.e. la détection conjointe des organes (tige, feuille) et de leur configuration spatiale. La principale innovation réside dans la capacité de ce réseau à détecter des structures ayant un nombre arbitraire d'organes. La performance de cette approche est ensuite comparée à la méthode de référence.

- Une proposition d'algorithme de suivi de points d'intérêt permettant d'augmenter la fiabilité des détections réalisées image par image en tirant parti de la redondance temporelle de la séquence d'images. Cet algorithme permet de lier de multiples observations de la même plante présentes dans plusieurs images successives afin d'augmenter la robustesse de la détection.

- L'acquisition et l'annotation d'une base de données d'images et de séquences d'images nécessaires à l'entraînement et l'évaluation des algorithmes d'apprentissage auto-

matique. Les conditions d'acquisition et les cultures supportées s'appuient sur le dispositif expérimental mis en place dans le projet BIPBIP, et plus généralement dans le Challenge ROSE.

La modélisation du problème et le détail des algorithmes développés sont présentés dans le chapitre 2, les protocoles d'évaluation, les expérimentations et l'évaluation des algorithmes sont étayés dans le chapitre 3.4 puis les conclusions et perspectives sont tirées dans le chapitre 3.5.

Chapitre 2

Localisation, suivi et analyse de structure de plantes

Sommaire

Ce chapitre détaille les contributions algorithmiques développées au cours de cette thèse. L'objectif est d'exploiter des algorithmes de vision par ordinateur et d'apprentissage profond pour localiser et suivre au cours du temps des organes de plantes dans des séquences d'images en couleur. La principale application étudiée est le désherbage de précision et la description de cultures en plein champ. Les méthodes doivent être adaptées aux contraintes d'utilisation en temps réel du bloc-outil de désherbage mécanique BIPBIP et à son mode de fonctionnement décrit dans la section 1.1.

Les contributions proposées s'articulent autour deux objectifs :

- D'une part, la détection de plantes et la caractérisation de plantes dans des images individuelles. L'objectif est de collecter des informations pertinentes pour le désherbage de précision et le suivi des cultures, telles que la position des tiges et des feuilles ainsi que le nombre de feuilles de chaque plante.

- D'autre part, l'amélioration de la performance de la détection en tirant parti de la redondance d'information présente dans les séquences d'images captées par le bloc-outil BIPBIP.

Les approches d'apprentissage profond sont explorées pour la tâche de localisation et de caractérisation de plantes dans les images. En effet, les capacités de généralisation des approches d'apprentissage profond permettent de mieux gérer la forte variabilité des conditions d'acquisition des images (type de terrain, niveau d'infestation d'adventices, etc.) et la variabilité des cultures elles-mêmes (type de plante d'intérêt, stade de développement, état sanitaire, etc.). Pour la tâche d'agrégation temporelle des détections, une approche plus classique est mise en œuvre puisque, ici, une approche par apprentissage profond requerrait une labellisation de données plus coûteuse nécessitant l'annotation de séquences d'images entières. Une méthode simplement basée sur une modélisation du mouvement du bloc-outil et des algorithmes de traitement d'images est choisie.

La section 2.1 pose le problème d'un point de vue formel. Elle introduit une modélisation de la plante et des caractéristiques que l'on cherche à prédire. Elle présente également les caractéristiques des images traitées et des séquences d'images ainsi que les concepts et notations employés dans ce chapitre.

Afin de comparer plusieurs approches d'apprentissage profond, deux méthodes sont proposées pour localiser et caractériser les organes de plantes dans les images :

- Un détecteur d'objets performant issu de la littérature, paramétré et entraîné de manière judicieuse pour réaliser des tâches de détection d'organes de plantes et d'estimation du nombre de feuilles. Cette approche est explorée dans la section 2.2.

- Une architecture neuronale originale dédiée à la détection et la caractérisation de plantes. Cette architecture permet de prédire la position des organes de plantes et leurs relations. Cette approche est présentée dans la section 2.3.

La section 2.4 détaille ensuite la méthode d'agrégation temporelle développée pour suivre au cours du temps la base des tiges des plantes d'intérêt afin d'augmenter la fiabilité de la détection. Enfin, une synthèse des méthodes produites est présentée dans la section 2.5.

2.1 Modélisation du problème

Cette section formalise les données du problème et présente les notations utilisées dans la suite. En particulier, elle introduit les modélisations de la plante, de l'image et de la séquence d'images sur lesquelles les méthodes proposées se fondent.

2.1.1 Modélisation de la plante

On propose de modéliser une plante comme une entité possédant un ensemble de propriétés qui la décrivent. Le choix des propriétés qui lui sont assignées influence grandement la manière d'annoter les images en vue d'un apprentissage automatique. Il est important de

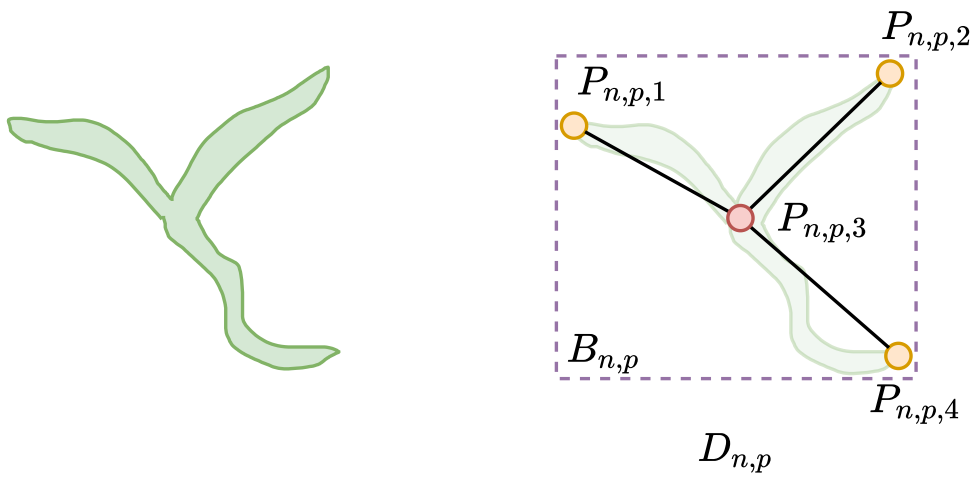


FIGURE 2.1 : Schéma d'une plante vue de dessus (à gauche) et de sa modélisation (à droite) $D_{n,p}$, où n est l'indice de l'image et p l'indice de la plante dans l'image. Les points de couleur identifient les points d'intérêt (organes) $P_{n,p}$ de la plante, avec en rouge le point supposé d'entrée de la tige dans le sol (potentiellement invisible à cause de masquages) et en jaune les pointes de feuilles. Le rectangle violet symbolise la boîte englobante $B_{n,p}$ de la plante entière.

choisir une modélisation à la fois simple, permettant de réduire le travail d'annotation et de décrire correctement la plante. Par exemple, un modèle basé sur une segmentation (classification au niveau du pixel) requiert l'annotation de chaque pixel, ce qui est fastidieux et chronophage. Au contraire, un modèle basé sur une boîte englobante (le rectangle le plus petit qui contient la plante, hors rotations) ou sur les points d'intérêt est rapide à annoter.

Les choix de modélisation réalisés s'appuient principalement sur les besoins et les contraintes du bloc-outil BIPBIP (voir section 1.1). En particulier, le bloc-outil a besoin de la position précise du point d'entrée des tiges de plantes dans le sol pour les éviter lors du binage. Une contrainte est que ce point d'entrée n'est pas forcément visible car les images sont prises à la verticale et que, dans cette configuration, il peut être masqué par les feuilles. Afin de satisfaire le besoin de description des cultures, un autre point d'intérêt est pris en compte : les feuilles des plantes. Cet indicateur permet notamment de réaliser du comptage de feuilles et donc de caractériser le stade de développement des cultures.

Le modèle choisi pour exprimer ces besoins est illustré dans la figure 2.1 par un schéma de la plante en vue de dessus (à gauche) et la modélisation choisie (à droite). L'idée est de définir des points d'intérêt non ambigus permettant de localiser précisément les organes de la plante : la base de la tige et les feuilles. Les points d'intérêt de base de tige (en rouge sur la figure) correspondent à la position estimée du point d'entrée de la tige dans le sol et les feuilles (en jaune sur la figure) sont identifiées par les pointes. Une plante est donc modélisée par un unique point d'intérêt de base de tige et plusieurs (zéro inclus) points d'intérêt de pointe de feuille. Cette modélisation inclut également la boîte englobante de la plante (en violet sur la figure) qui permet de caractériser l'extension spatiale de la plante

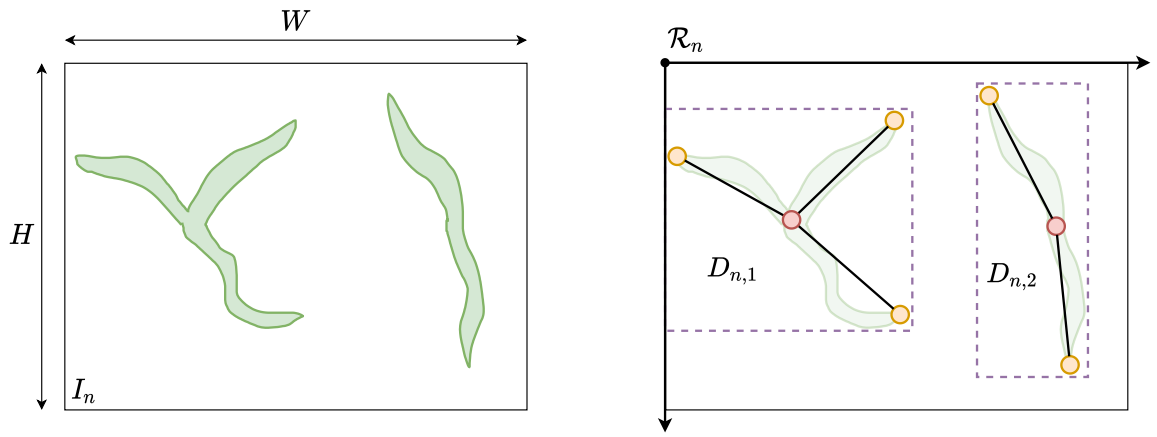


FIGURE 2.2 : Schéma de la modélisation d'une image I_n (à gauche) et illustration d'objets de plantes D_n (à droite) avec la boîte englobante (violet) et les points d'intérêt d'organes de tige (rouge) et de feuille (jaune). Les coordonnées des différentes propriétés sont exprimées dans le référentiel $\mathcal{R}_n$ de l'image (matérialisé par les flèches noires).

et la variété de la plante (maïs, haricot, etc.).

Ce modèle n'est pas parfait : il ne tient ainsi pas compte de la forme des feuilles qui peuvent ne pas être droites ou de la présence de feuilles secondaires accrochées à des tiges secondaires. Certaines feuilles peuvent aussi masquer d'autres feuilles et rendre invisibles certains points d'intérêt. La position exacte n'est donc en général pas connue et est seulement estimée. Ce modèle est néanmoins pertinent au stade de développement précoce envisagé.

Plus formellement, une plante visible dans une image $I_n,\ n \in \mathbb{N}^+$ est caractérisée par un ensemble de propriétés $D_{n,p}$ où $p \in \mathbb{N}^+$ est l'indice de la plante dans l'image, qui peut en contenir plusieurs. L'indice n de l'image correspond à *l'instant* de la prise de vue, i.e. l'image I_{n+1} est capturée à un instant $n+1$ ultérieur à celui de l'image I_n. On définit un *objet* $D_{n,p}$ représentant les propriétés d'une plante donnée de l'image par :

$$D_{n,p} = \{B_{n,p},\ P_{n,p},\ c_{n,p}\}\,, \tag{2.1}$$

où $B_{n,p}$ est la boîte englobante de la plante, $c_{n,p}$ un label qui identifie l'espèce de la plante (maïs, haricot, etc.) et $P_{n,p}$ est l'ensemble des points d'intérêt de la plante :

$$\begin{aligned} P_{n,p} &= \{P_{n,p,j}\}_{j \in 1 \ldots N_{n,p}} \\ &= \left\{P^x_{n,p,j},\ P^y_{n,p,j},\ P^c_{n,p,j}\right\}_{j \in 1 \ldots N_{n,p}}, \end{aligned} \tag{2.2}$$

où $P_{n,p,j}$ est un point d'intérêt défini par sa position $(P^x_{n,p,j},\ P^y_{n,p,j})$ dans l'image et par son type $P^c_{n,p,j}$ (feuille ou base de tige). Finalement, $N_{n,p}$ correspond au nombre de points d'intérêt de la plante $D_{n,p}$, donc le nombre de feuilles plus un.

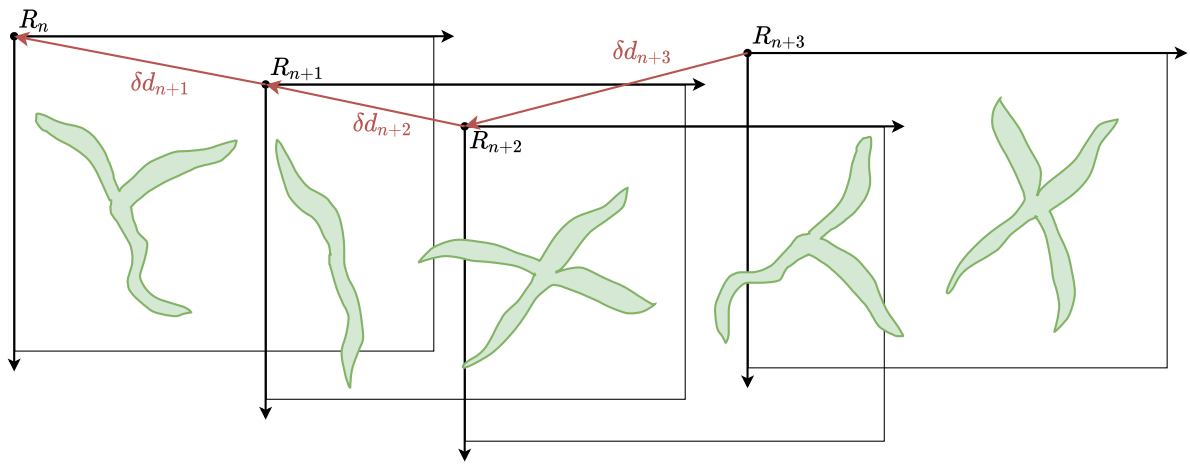


FIGURE 2.3 : Schéma de la modélisation d'une séquence de quatre images consécutives captées par le bloc vision du module BIPBIP. Les images se recouvrent, ce qui entraîne une redondance spatiale. Le déplacement entre deux images est matérialisé par un vecteur rouge.

L'ensemble des plantes d'une image I_n donnée est défini comme :

$$D_n = \{D_{n,p}\}_{p \in 1 \ldots N_n}, \tag{2.3}$$

où N_n est le nombre de plantes dans l'image I_n. Toutes les coordonnées sont exprimées dans le référentiel $\mathcal{R}_n$ de l'image I_n qui prend pour origine le coin en haut à gauche de l'image. La figure 2.2 illustre ce référentiel ainsi que plusieurs objets D_n dans une image I_n. Toutes les images ont la même définition $W \times H$ où W est la largeur et H la hauteur.

Pour résumer, une plante est modélisée par un *objet* qui regroupe un ensemble de propriétés décrivant la plante : un *label* qui identifie sa variété, une *boîte englobante* qui localise l'objet entier dans l'image et un ensemble de *points d'intérêt* comprenant une base de tige et, éventuellement, une ou plusieurs pointes feuilles. Une *prédiction* (ou une *détection*) désigne une estimation de ces paramètres réalisée par un algorithme donné.

2.1.2 Modélisation de la séquence d'images

La configuration du module de vision du bloc-outil BIPBIP présenté dans la section 1.1 contraint les prises de vues de la caméra couleur. Celle-ci étant positionnée à la verticale et à une distance fixe du sol. Lors du déplacement du bloc-outil le long du rang, la caméra capte des images des cultures à une certaine fréquence, ce qui occasionne de larges recouvrements entre images successives. C'est cette redondance qui peut être exploitée pour améliorer les détections réalisées à l'échelle de l'image. Le schéma de la figure 2.3 illustre ce procédé sur quatre images consécutives d'un rang.

L'hypothèse faite est qu'en connaissant le déplacement de la caméra δd_{n+1} dans le référentiel lié au sol entre deux images successives I_n et I_{n+1} (matérialisé par les vecteurs

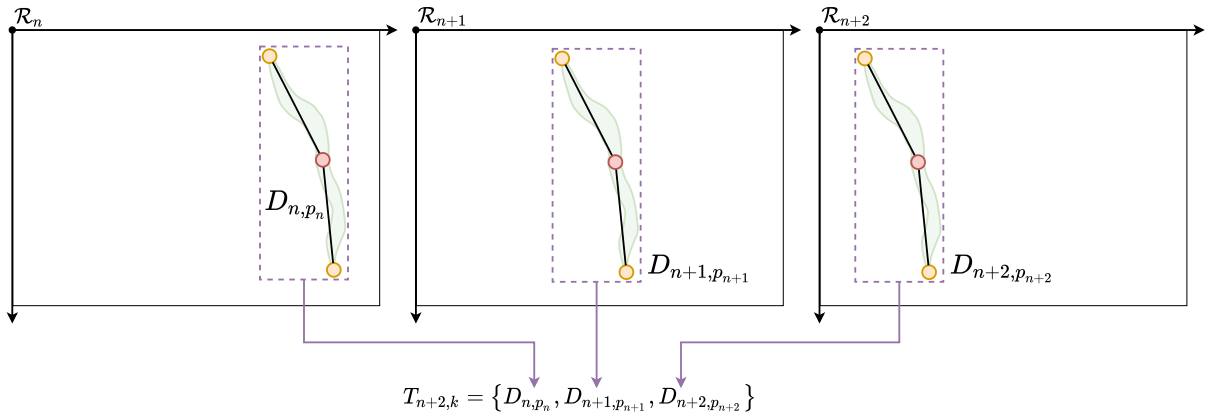


FIGURE 2.4 : Schéma de la modélisation d'un tracker. Le tracker $T_{n+2,k}$ représente la même plante sous-jacente détectée plusieurs fois dans trois images consécutives.

en rouge sur la figure), il est possible de calculer le déplacement des points d'intérêt d'ancrage au sol, comme les bases de tige de plante, et ainsi d'associer les points d'intérêt qui représentent la même base de tige dans les deux images. C'est donc le point d'intérêt de tige qui permet d'identifier et de regrouper les objets qui représentent la même plante dans plusieurs images. Cette hypothèse est étayée avec plus de détails dans la section 2.4.

La notion d'identité de plante est formalisée par le concept de *tracker*, illustré dans le schéma de la figure 2.4. Un tracker regroupe un ensemble de détections provenant d'images différentes qui représentent la même plante : il y a donc un tracker pour chaque plante observée. Les algorithmes développés sont voués à être utilisés en temps réel et doivent donc être causaux : une prédiction à un instant n ne doit dépendre que des prédictions des instants passés $D_m,\ m \leq n$. Chaque tracker est défini à l'instant n et doit être "mis à jour" que lorsque la prédiction suivante est disponible. Ce processus itératif de mise à jour des trackers est désigné par *agrégation* dans la suite. Un tracker k à l'instant n est défini comme :

$$T_{n,k} = \{D_{n_i,p_i}\}_{i \in 1 \ldots N_k}\ ,\ n_i \in 1 \ldots n,\ p_i \in 1 \ldots N_{n_i}, \tag{2.4}$$

où N_k est le nombre d'observations de la plante pour le tracker k, N_{n_i} le nombre de plantes présentes dans l'image I_{n_i}, D_{n_i,p_i} est l'une des N_k occurrences de la plante. La figure 2.4 illustre un tracker T_{n+2} constitué de trois détections de plante provenant d'images consécutives représentant la même plante. Finalement, l'ensemble des trackers T_n à l'instant n est :

$$T_n = \{T_{n,k}\}_{k \in 1 \ldots N_n^t} \tag{2.5}$$

où N_n^t est le nombre de trackers[1].

Pour résumer, un tracker représente l'identité stable d'une plante : les objets d'images

[1]En théorie, le nombre de trackers N_n^t est égal au nombre total de plantes observées dans la séquence d'images $\{I_i\}_{i \in 1 \ldots n}$. En pratique, lors de la prédiction, le nombre de trackers $\hat{N}_n^t$ peut être supérieur ou inférieur à N_n^t à cause d'erreurs de prédiction.

différentes qui représentent la même plante appartiennent au même tracker. Les objets d'un tracker sont associés via un processus d'agrégation basé sur l'estimation du déplacement de la tige de la plante entre images successives. L'agrégation est donc un processus itératif causal qui met à jour l'ensemble des trackers en cours de suivi.

Bilan Les plantes sont modélisées par un ensemble de propriétés qui les décrivent : la localisation et le type d'organe, le nombre de feuilles, la variété de la plante ou encore la boîte englobante. Chaque image I_n contient un certain nombre de plantes et l'objectif des approches d'apprentissage profond développées est d'estimer les objets D_n qui représentent ces plantes. Les images se présentent sous la forme de séquences qui se recouvrent spatialement, introduisant une redondance d'information qui est exploitable pour augmenter la qualité de la prédiction réalisée par les algorithmes d'apprentissage profond. Pour cela, le mouvement de la caméra entre deux images successives doit être estimé afin d'associer les objets qui représentent la même plante. Cette association, appelée agrégation, est guidée par la tige des plantes qui permet de les identifier de manière unique et de les regrouper au sein d'un tracker.

Les sections suivantes détaillent deux méthodes d'estimation des propriétés de plantes basées sur de l'apprentissage profond, l'une via un réseau de neurones de détection d'objets présentée dans la section 2.2, et l'autre via une architecture dédiée à la détection de structure de plante détaillée dans la section 2.3. Un algorithme de vision par ordinateur qui implémente un processus d'agrégation temporelle des tiges de plante est ensuite présenté dans la section 2.4.

2.2 Localisation d'organes et comptage du nombre de feuilles via un réseau de neurones de détection d'objets

Cette section présente une méthode d'apprentissage profond basée sur un réseau de neurones de détection d'objets pour localiser et caractériser les organes de plantes. Le but est d'exploiter de façon judicieuse des architectures déjà existantes et bien maîtrisées, performantes et suffisamment flexibles pour résoudre le problème adressé.

La première section 2.2.1 présente la manière dont les détecteurs d'objets peuvent être exploités pour la localisation de points d'intérêt et la caractérisation de plantes. La deuxième section 2.2.2 discute le compromis entre la précision de la détection et la rapidité d'inférence de ces réseaux. La dernière section 2.2.3 introduit l'architecture sélectionnée pour résoudre le problème.

2.2.1 Détection d'objets et potentiel pour la localisation d'organes et la caractérisation de plantes

Cette section présente les concepts de la détection d'objets ainsi que les potentialités d'utilisation pour localiser des points d'intérêt comme les organes de plante et caractériser des plantes, par exemple par le nombre de feuilles qu'elle possède.

2.2.1.1 Détection d'objets

Dans le domaine de l'apprentissage profond, un détecteur d'objet désigne généralement une architecture neuronale capable de prédire un ensemble de boîtes rectangulaires, nommées *boîtes englobantes*, qui englobent les objets d'intérêt au plus près de leurs limites spatiales, i.e. les bords de l'objet. Chaque boîte englobante est associée à un *label* (ou *classe*) qui désigne le type d'objet contenu dans la boîte. Par exemple, un réseau de neurones de détection de véhicules prédit des boîtes englobantes qui correspondent à un label pouvant être "voiture", "vélo" ou "camion".

Les boîtes englobantes rectangulaires ont une orientation fixe, les côtés étant parallèles aux axes de l'image. Une boîte englobante peut être décrite avec quatre coordonnées : la position du centre de la boîte dans le référentiel de l'image $(x_{n,i}, y_{n,i})$ et la taille de la boîte $(w_{n,i}, h_{n,i})$, où $x_{n,i}$ est la position sur l'axe horizontal, $y_{n,i}$ la position sur l'axe vertical, $w_{n,i}$ la largeur et $h_{n,i}$ la hauteur de la boîte, n désigne l'indice de l'image et i l'indice de la boîte dans l'image n. Un réseau de neurones de détection d'objets génère donc pour l'image I_n un ensemble de détections d'objets D_n qui s'exprime comme :

$$D_n = \{(x_{n,i}, y_{n,i}, w_{n,i}, h_{n,i}, c_{n,i})\}_{i \in 1 \ldots N_n} , \tag{2.6}$$

où $c_{n,i}$ est le label de la boîte englobante $B_{n,i} = (x_{n,i}, y_{n,i}, w_{n,i}, h_{n,i})$ et N_n le nombre de boîtes englobantes dans l'image I_n. La figure 2.5 illustre un exemple de détection de plante $D_{n,i}$ avec la boîte englobante et la classe $c_{n,i}$. En résumé, une détection d'objet consiste en l'association d'une zone rectangulaire à un label décrivant ce qui est présent dans la zone.

2.2.1.2 Potentiel pour la localisation de points d'intérêt et la caractérisation d'objets

Les labels utilisés en détection d'objets décrivent usuellement des objets significativement différents. Par exemple, un usage approprié en détection de cultures maraîchères serait d'utiliser des labels tels que "maïs", "haricot" et "poireau" pour décrire plusieurs espèces différentes. Cependant, rien n'empêche de permettre plus de granularité. Il est tout à fait possible de différencier plusieurs variétés d'une espèce. Il serait envisageable, par exemple, de détecter trois variétés de maïs, deux de haricot et une de poireau. Il est donc parfaitement possible d'utiliser des labels qui décrivent à la fois l'objet et certains

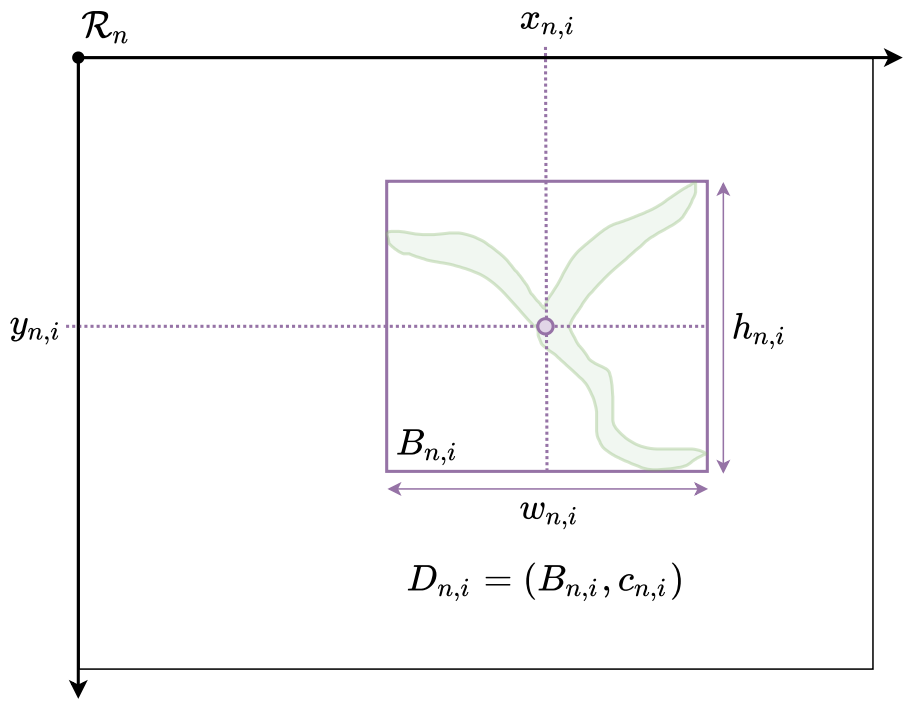


FIGURE 2.5 : Schéma du modèle de la boîte englobante utilisé en détection d'objets. Une boîte englobante $B_{n,i}$ est paramétrée par la position de son centre $(x_{n,i}, y_{n,i})$ et par sa taille $(w_{n,i}, h_{n,i})$ exprimées dans le référentiel $\mathcal{R}_n$ de l'image I_n. Une détection $D_{n,i}$ est paramétrée par sa boîte englobante $B_{n,i}$ et son label $c_{n,i}$.

attributs de celui-ci. Ainsi, des attributs de plantes possibles pourraient être le stade de développement, l'orientation, le nombre de feuilles, la présence de maladie ou non, le besoin en eau, etc.

La seule limitation réside dans le fait que l'attribut soit "encodé" dans le label. Or, les labels et leur nombre sont fixés avant l'entraînement et ne peuvent pas êtres modifiés ultérieurement. Les labels, et donc les attributs, représentent nécessairement une variable nominale, c'est-à dire une variable discrète et non-ordonnée. Nous montrerons dans ce travail un exemple d'utilisation d'un réseau de détection où une information non triviale, telle que le nombre de feuilles, est encodée dans le label de la boîte. Ce nombre est un indicateur important pour décrire le stade de développement d'une culture.

Localiser des points d'intérêt via un réseau de neurones de détection d'objets est peu évident puisqu'ils ne sont a priori pas conçus pour de telles applications. Un point d'intérêt est un point dans l'espace associé à un label. Une méthode de détection de points d'intérêt a pour but la prédiction d'un ensemble de points :

$$K_n = \{(x_{n,i}, y_{n,i}, c_{n,i})\}_{i \in 1 \ldots N_n} , \tag{2.7}$$

où $c_{n,i}$ est le label du point, par exemple "tige" ou "feuille" en détection de plantes. Il faut cependant remarquer que cette définition est proche de celle de la détection d'objets

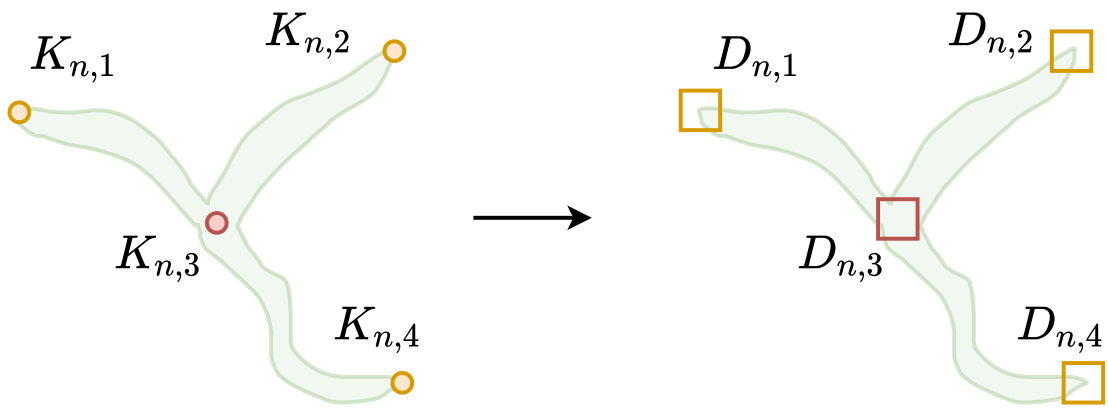


FIGURE 2.6 : Schéma de la méthode de conversion des points d'intérêt en boîtes englobantes illustrée pour la détection d'organes de plantes. Les points d'intérêt sont illustrés par des points (à gauche) et les boîtes par des carrés (à droite).

présentée dans l'équation 2.6, i.e. une boîte englobante est un point d'intérêt auquel est ajoutée la notion de taille[2]. Une approche possible pour utiliser un détecteur d'objets comme détecteur de points d'intérêt est donc simplement d'ignorer le paramètre de taille des boîtes.

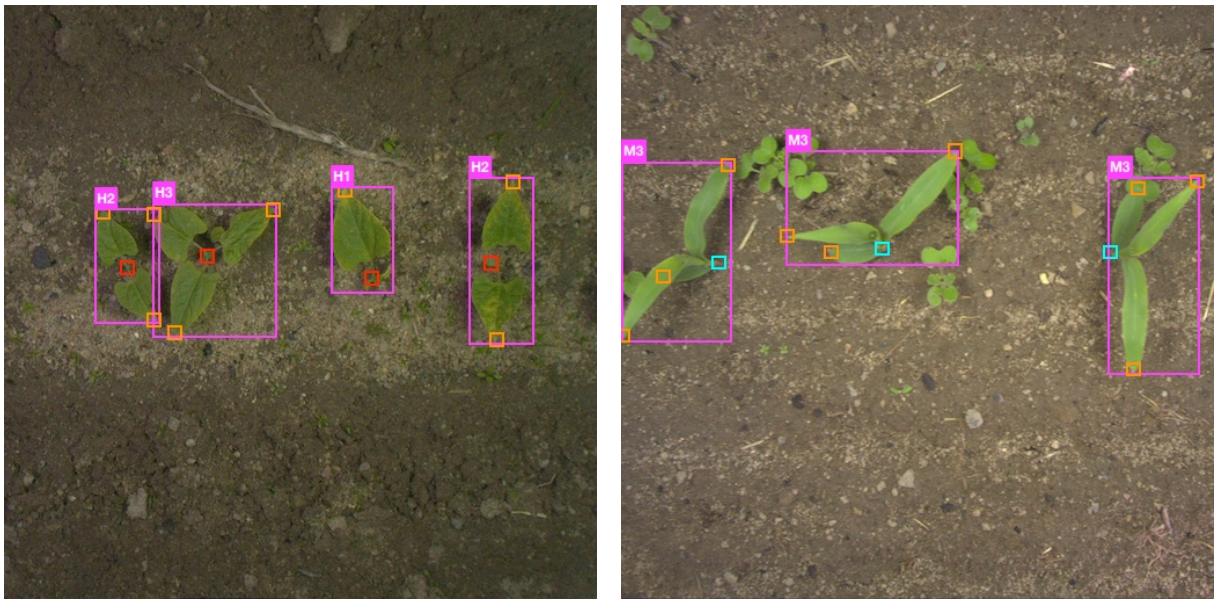
Néanmoins, pour entraîner un détecteur d'objets, il faut nécessairement utiliser une formulation similaire à celle de l'équation 2.6, il faut donc définir ce qu'est la représentation "boîte englobante" d'un point d'intérêt afin de convertir les points d'intérêt en boîtes englobantes pour l'entraînement. Une solution qui satisfait ce prérequis est de modéliser un point d'intérêt par une boîte englobante dont le centre correspond à la position du point et les paramètres de tailles sont fixés à une certaine valeur, i.e. un point d'intérêt est représenté par une boîte englobante carrée centrée sur celui-ci :

$$D_n = \{(x_{n,i}, y_{n,i}, L_b, L_b, c_{n,i})\}_{i \in 1 \ldots N_n} , \tag{2.8}$$

avec $L_b \in \mathbb{N}^+$ la longueur des côtés de la boîte carrée et $c_{n,i}$ le label du point d'intérêt. La figure 2.6 illustre cette modélisation pour les points d'intérêt d'organes de plantes (tige et feuilles) : les points sont transformés en boîtes englobantes carrées et centrées sur les points d'intérêt.

Pour résumer, il est possible d'exploiter un réseau de neurones de détection d'objets pour caractériser un objet en encodant les attributs dans le label des boîtes englobantes. Cela limite néanmoins la plage de valeurs possibles des attributs puisque le nombre de labels doit être fini. Il est également possible de réaliser de la localisation de points d'intérêt en utilisant des boîtes englobantes centrées sur les points. Cette formulation implique de fixer la taille de la boîte englobante à une valeur arbitraire, ce qui peut influencer la

[2]C'est littéralement la manière de procéder de la famille des réseaux de détection d'objets par points [33, 80, 142, 143] : un objet est modélisé par un point puis sa taille est apprise comme un attribut de ce point.



(a) Haricot (b) Maïs

FIGURE 2.7 : Illustration de la détection d'organes et d'estimation du nombre de feuilles du haricot et de maïs via un réseau de neurones de détection d'objets. Les bases de tiges sont détectées par des boîtes englobantes centrées sur le point d'entrée dans le sol (boîtes rouges pour le haricot et bleues pour le maïs) et les feuilles par des boîtes centrées sur la pointe (en orange). Le nombre de feuilles de chaque plante est estimé par le label de la boîte englobante de plante entière (boîtes roses) qui encode à la fois la variété de la plante ("H" pour le haricot, "M" pour le maïs) et le nombre de feuilles (indiqué par le nombre à côté de la lettre).

performance du réseau et doit donc être validé par des expérimentations. La figure 2.7 illustre la détection d'organes de plantes et l'estimation du nombre de feuilles sur des plants de haricot et de maïs. Les bases de tige sont détectées par des boîtes englobantes centrées sur le point d'entrée dans le sol (boîtes rouges pour le haricot et bleues pour le maïs) et les feuilles par des boîtes centrées sur la pointe (en orange). Le nombre de feuilles de chaque plante est estimé par le label de la boîte englobante de plante entière (boîtes roses) qui encode à la fois la variété de la plante ("H" pour le haricot, "M" pour le maïs) et le nombre de feuilles (indiqué par le nombre à côté de la lettre).

Les sections suivantes présentent les paramètres clés à prendre en compte dans le choix d'un réseau de neurones de détection d'objets puis motivent le choix d'une architecture vis-à-vis des contraintes et spécifications du bloc-outil BIPBIP.

2.2.2 Le compromis vitesse-précision des réseaux de neurones de détection d'objets

Il y a principalement deux métriques employées dans la littérature pour évaluer les performances des réseaux de neurones de détection d'objets :

- **La précision**[3] des prédictions, c'est-à-dire la capacité du réseau à prédire correctement la position des boîtes englobantes, leur taille et leur label. Pour la détection d'objets, la précision est couramment mesurée par l'AP [105] (AP signifie "Average Precision", voir section 3.1). Elle dépend principalement du type de réseau, de sa profondeur et de la méthode d'entraînement du réseau.

- **La vitesse d'inférence** qui est la fréquence à laquelle le réseau de neurones est capable de traiter les images. Elle est mesurée par le nombre d'images par secondes (IPS). La vitesse d'exécution est évidemment dépendante du système qui exécute le réseau, autant du matériel (CPU, GPU, TPU, etc.) [131] que des librairies de calcul disponibles (CUDA, TensorRT, etc.) [121].

Il y a d'autres métriques d'intérêt mais elles sont moins impactantes en pratique. C'est le cas de l'empreinte mémoire du réseau (la taille de la mémoire vive ou de la mémoire graphique sont généralement suffisantes pour la majorité des réseaux).

Ces deux métriques sont liées par une relation dite de *compromis vitesse-précision* (Speed-Accuracy Trade-Off), l'amélioration de l'une de ces métriques se faisant souvent au détriment de la seconde. En effet, la précision de la prédiction d'un réseau est corrélée à la profondeur du réseau [126] (et plus globalement au nombre de neurones). Or un réseau plus profond est plus coûteux calculatoirement puisqu'il y a plus d'opérations à réaliser. Un réseau plus précis est donc généralement plus lent. Il y a bien sûr des exceptions à cette règle et elle n'est valide que pour les réseaux d'une même "génération" (réseaux de la même famille ou utilisant une architecture similaire). Entre autres, un tout nouveau détecteur d'objets est en principe meilleur que les réseaux existants sur les deux métriques à la fois, ou a minima à égalité sur l'une et meilleur sur l'autre. Le choix d'un réseau de détection dépend donc des besoins en précision et en vitesse d'exécution de l'application visée. La figure 2.8 illustre ce compromis sur un graphique de la précision en fonction de la vitesse d'inférence. Les réseaux en bleu représentent l'état de l'art actuel, les réseaux en rouge sont obsolètes et ceux en vert représentent les futurs réseaux plus performants. La courbe en pointillés représente l'optimum de Pareto[4].

Néanmoins, comparer équitablement les différents réseaux de la littérature entre eux n'est pas toujours évident [87, 59] pour plusieurs raisons :

- Il est difficile d'identifier la source des gains de performances annoncés. Les réseaux sont souvent vus comme des boîtes noires difficilement interprétables. Les gains de performances sont rarement prouvés et relèvent souvent de la spéculation. Par exemple, l'article original sur la normalisation de batch [62] (*Batch Normization* (BN)), qui améliore grandement la rapidité de convergence de l'entraînement, faisait

[3] À ne pas confondre avec la mesure statistique éponyme, introduite dans la section 3.1.

[4] L'optimum de Pareto définit une situation où une amélioration sur un aspect (par exemple la vitesse d'inférence) se fait au détriment d'un autre aspect (par exemple la précision).

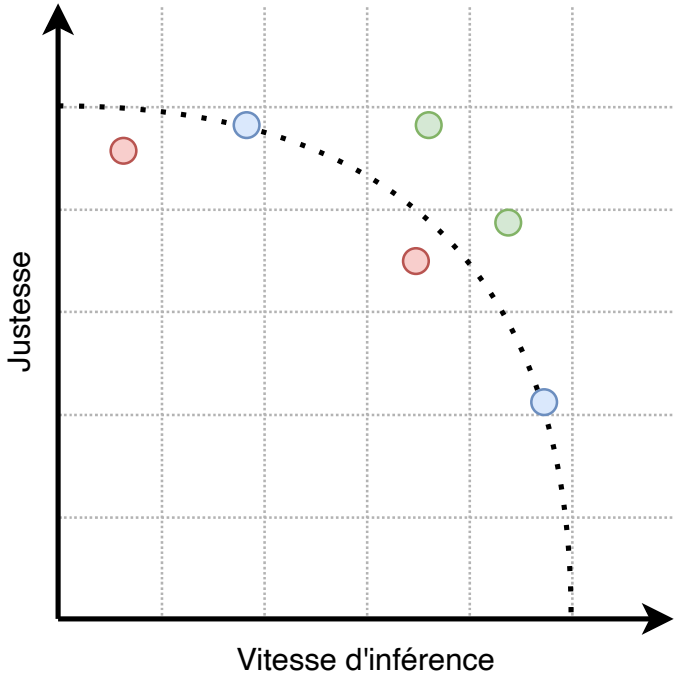


FIGURE 2.8 : Illustration simplifiée du compromis vitesse-précision pour les réseaux de neurones de détection d'objets. La courbe en pointillés illustre l'optimum de Pareto et chaque point de couleur illustre une architecture spécifique. Les réseaux en bleu sont les réseaux optimaux de l'état de l'art, ceux en rouge sont les réseaux obsolètes et ceux en vert les futurs "meilleurs" réseaux. Les deux réseaux en bleu illustrent des compromis différents.

l'hypothèse que cette amélioration était due à la réduction du "décalage de covariable interne" (*Internal Covariate Shift*). Trois ans plus tard, un autre article [118] invalide cette théorie et démontre que la raison n'est pas à chercher du côté du décalage de covariable mais de la régularisation de la fonction de coût.

- Il est difficile d'éliminer les facteurs de confusion sur l'origine du gain de performance. Une nouvelle architecture présente souvent plusieurs améliorations distinctes en même temps, par exemple une nouvelle couche, des connexions différentes, une meilleure méthode d'entraînement et parfois même de nouvelles données d'apprentissage. Malgré l'intérêt des études d'ablation (*Ablation Studies*) qui cherchent à réduire les facteurs de confusion en n'étudiant qu'un paramètre à la fois, le caractère universel des gains obtenus par telle ou telle modification est rarement garanti : un article peut prétendre qu'une méthode améliore les performances tandis qu'un autre article qui implémente cette méthode sur une autre architecture ne note pas de gain notable. Par exemple l'article sur le FocalLoss [85] présente une nouvelle fonction de coût améliorant grandement la précision d'un réseau de détection. Un an plus tard, un autre article [113] présente une "meilleure" architecture, mais montre qu'utiliser la nouvelle fonction de coût FocalLoss baisse significativement les performances du

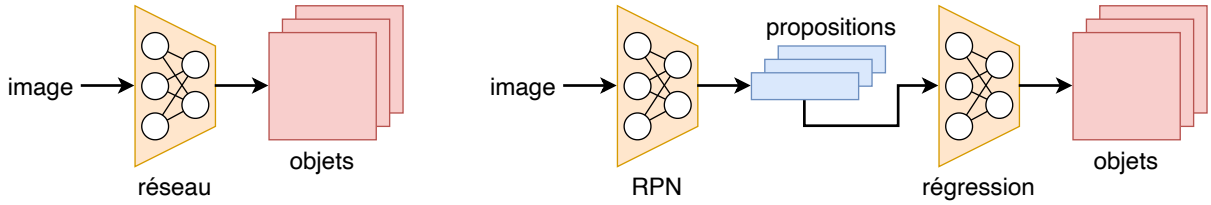


FIGURE 2.9 : Illustration simplifiée des deux catégories de réseaux de neurones de détection d'objets : les réseaux à un étage (à gauche) et les réseaux à deux étages (à droite). Les réseaux à deux étages réalisent d'abord une prédiction de boîtes englobantes potentielles via un réseau de proposition de régions (RPN) puis régressent la position, la taille et la classe. Les réseaux à un étage prédisent directement les boîtes avec leur classe.

réseau utilisé.

De plus, il faut mentionner que le code de calcul des performances de précision (AP) est parfois différent selon les articles et ne donne pas les mêmes résultats d'un ordinateur à l'autre [105] ou que la vitesse d'inférence est calculée sur des ordinateurs différents et avec des optimisations (CUDA, TensorRT) différentes selon les articles. En résumé, il est difficile de comparer équitablement les différentes architectures neuronales à moins de les ré-entraîner soi-même dans un environnement contrôlé et de les évaluer dans les conditions réelles de l'application visée, ce qui représente une quantité non négligeable de travail et des coûts d'entraînement substantiels.

Malgré ces remarques, il est tout de même possible d'identifier plusieurs considérations architecturales qui influencent fortement la précision et la vitesse d'inférence. De nombreux travaux [66, 69, 88, 144, 59] analysent les caractéristiques les plus influentes sur ces deux paramètres. En l'occurrence, deux grandes catégories de réseaux de détection d'objets illustrées dans la figure 2.9 sont observées :

- **Les réseaux à deux étages** dits *two-stage* (à droite dans la figure), qui sont historiquement les premiers à avoir été utilisés pour la détection d'objets [66]. Deux réseaux de neurones successifs, appelés *étages*, sont employés : le premier est un réseau de proposition de régions (*Region Proposal Network*), i.e. qui propose des boîtes englobantes potentielles. Le second régresse la position et le label des boîtes englobantes. On trouve dans cette catégorie des réseaux comme R-CNN (*Region Convolutional Neural Network*) [44], ensuite amélioré par Fast R-CNN [43] et Faster R-CNN [114]. Ces réseaux sont meilleurs sur la métrique de précision, au détriment de la métrique de vitesse d'inférence.

- **Les réseaux à un étage** dits *one-stage* (à gauche dans la figure), qui n'ont qu'un seul étage chargé de la génération des boîtes englobantes et des labels. On trouve dans cette catégorie RetinaNet [85], les réseaux SSD (*Single-Shot Multibox Detector*) [39, 90, 117] et la famille de réseaux YOLO (*You Only Look Once*) [13, 41, 111, 112, 113, 132]. Ces réseaux sont moins performants sur la métrique de précision mais plus rapides,

et souvent les seuls à pouvoir être employés pour une utilisation en temps réel.

Ce constat permet d'orienter le choix sur une catégorie plutôt qu'une autre selon l'application visée. Une application qui requiert une bonne précision de prédiction mais qui ne nécessite pas de décision en temps réel (par exemple, une application qui calcule un indicateur de maladie ou de croissance d'une culture utilisé pour assister une décision humaine sur le long terme) se tournera vers des réseaux à deux étages. Si une bonne vitesse d'exécution est néanmoins requise, l'utilisation d'un serveur local de calcul performant ou d'une exécution sur le réseau (*Cloud Computing*) compensera le manque. En revanche, une application nécessitant une action en temps réel (par exemple une application en robotique agricole pour du désherbage, de la pulvérisation ou de l'irrigation de précision) préférera un réseau à un étage, quitte à choisir une variante du réseau plus lente (voir section suivante 2.2.3) si la précision n'est pas suffisamment bonne, ou à changer de matériel de calcul pour une solution plus rapide.

Pour résumer, les deux métriques principales pour évaluer la performance des réseaux de neurones de détection d'objets sont la précision de la prédiction et la vitesse d'inférence. Au sein d'une même famille de réseaux, ces quantités évoluent de manière opposée : augmenter la vitesse d'inférence réduira la précision de la prédiction, c'est ce que l'on appelle le compromis vitesse-précision. Évaluer objectivement ces deux métriques pour plusieurs architectures n'est pas évident car identifier la cause des gains de performance est généralement difficile et, de plus, l'environnement d'exécution[5] des réseaux diverge beaucoup d'une architecture à une autre. Le choix d'une catégorie de réseau (un ou deux étages) repose sur des considérations applicatives et est guidé par la répartition des besoins sur les deux métriques d'évaluation des performances de réseaux. Le choix de la famille (famille YOLO v.s. famille SSD par exemple) repose plus sur des considérations techniques. Par exemple une application "prête pour la production" se tournera vers des familles de réseaux proposant une implémentation très optimisée et éprouvée, alors qu'une application plus prospective préférera des implémentations plus flexibles et reproductibles.

La section suivante présente les contraintes techniques et les motivations qui ont guidé le choix d'une catégorie et d'une famille de réseaux de neurones de détection d'objets.

2.2.3 YOLO : une architecture adaptée à la détection en temps réel

Le choix d'un réseau de neurones de détection d'objets issu de la littérature dépend des besoins et des contraintes d'intégration. Dans le cadre du projet BIPBIP, il y a une contrainte d'exécution en temps réel à respecter (environ 10 images par seconde). La puissance disponible est aussi limitée et l'ordinateur embarqué (voir la section 1.1)

[5]Autant l'environnement matériel (type et famille de GPU, coprocesseurs de tenseurs (TPU) et autres accélérateurs) que l'environnement logiciel (CUDA, CuDNN, TensorRT, précision mixte, etc.) peuvent avoir un impact significatif sur la vitesse d'inférence et la précision d'un réseau.

doit pouvoir exécuter d'autres algorithmes (décision, commande des outils de binage) en parallèle du réseau de neurones. De plus, dans l'optique d'une future commercialisation du bloc-outil BIPBIP, une architecture neuronale qui a fait ses preuves et correctement optimisée sera préférée à une architecture plus expérimentale.

Ces considérations ont orienté le choix sur la catégorie des réseaux à un seul étage, ceux-ci étant les plus rapides. Le choix de la famille de réseaux est plus subjectif, selon une étude datant de 2019 [66], les familles de détecteurs d'objets à un étage évalués sont presque à égalité sur la métrique de précision mais les vitesses d'inférence varient grandement. Il est difficile de départager davantage les familles : l'environnement de test variant beaucoup (optimisations de calcul, etc.) et certains paramètres des réseaux ne sont pas comparables (notamment la définition des images en entrée des réseaux qui ne sont pas les mêmes).

Une autre étude plus récente [131] compare plusieurs familles de réseaux à un étage de la manière la plus équitable possible (même taille d'image en entrée, même matériel de calcul, mêmes optimisations, etc.) et montre des résultats relativement proches de la limite de Pareto (courbe de la valeur optimale du couple vitesse-précision) malgré des compromis différents (certaines familles favorisent la vitesse d'exécution, d'autres la précision de la prédiction). Par exemple les réseaux de la famille CenterNet [142] sont plus justes alors que les réseaux de la famille YOLO (variante Tiny par exemple) [113] sont plus rapides et ceux de la famille SSD [117] sont plus équilibrés. Cette étude montre aussi les différents compromis vitesse-précision possibles au sein d'une même famille (YOLOv3 v.s. YOLOv3 Tiny et CNet(D34) v.s. CNet(R101) par exemple, le premier étant plus lent mais plus juste).

Il ressort de ces études que les principaux réseaux de détection d'objets de l'état de l'art sont plus ou moins comparables sur les deux métriques d'évaluation, et, que de toute manière, une large gamme de compromis vitesse-précision est possible au sein d'une même famille. Plus subjectivement, la famille de détecteurs à un étage YOLO possède quelques avantages intéressants : (i) la base de code, Darknet[6], est activement maintenue, continue d'évoluer en tant que projet open-source et possède une large communauté active, et (ii) les optimisations sont transparentes (notamment pour les plateformes embarquées comme le NVIDIA Jetson Xavier). Cela permet d'accélérer le développement et l'intégration de nouveaux réseaux et de bénéficier des nouvelles avancées. Ainsi, une nouvelle variante de YOLO plus performante, YOLOv4 [13], a récemment été intégrée au framework Darknet. En pratique, le framework Darknet permet d'entraîner une très large gamme de réseaux YOLO (du premier YOLOv1 [111] au dernier YOLOv4 [13]), d'évaluer plusieurs variantes (variantes "Tiny" plus rapides mais moins précises, etc.) et d'utiliser ces réseaux sur du matériel de calcul embarqué, le tout dans un environnement unifié permettant une bonne homogénéité des résultats.

[6] `https://github.com/AlexeyAB/darknet/`

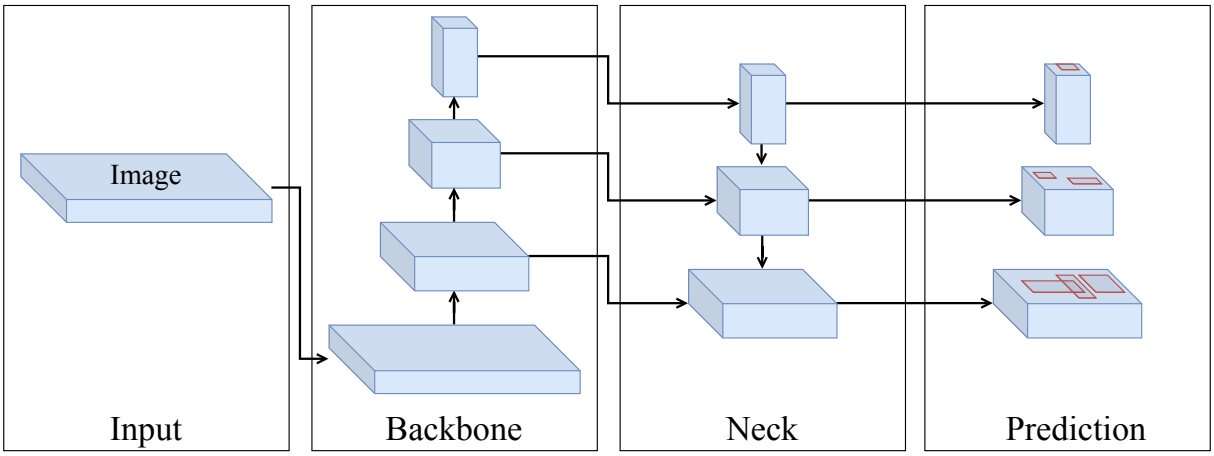


FIGURE 2.10 : Schéma simplifié du réseau de neurones de détection d'objets YOLOv4. La colonne extrait un ensemble de descripteurs, la nuque agglomère l'information sur plusieurs échelles et la dernière couche réalise les prédictions, une par échelle. Le haut des pyramides correspond à une échelle large (définition spatiale plus faible) et le bas à une échelle fine (définition spatiale plus fine).

La figure 2.10 présente un schéma simplifié de l'architecture neuronale de YOLOv4. La colonne (*backbone*) est un réseau de neurones qui agit comme un extracteur de descripteurs, ces descripteurs forment une pyramide où l'information est de plus en plus abstraite mais aussi de moins en moins localisée (perte de détail spatial). Dans le cas de YOLOv4, la colonne est un réseau convolutif par résidus (*ResNet* [56]) basé sur Darknet-53 [113]. La nuque (*neck*) est un réseau de neurones permettant de construire une pyramide de descripteurs multi-échelles où l'information est à la fois hautement abstraite et bien localisée. La nuque de YOLOv4 est basée sur le Path Aggregation Network (PAN) [89], qui est une variante de la nuque Feature Pyramid Network (FPN) [84]. La couche de prédiction (à droite de la figure) contient plusieurs têtes de régression (*prediction head*), qui sont des couches convolutives basiques, une par échelle.

En résumé, la famille de réseaux YOLO présente d'excellentes performances similaires aux autres détecteurs d'objets à un étage de la littérature. Le framework Darknet permettant d'entraîner cette famille de réseaux et de les intégrer sur des systèmes de calcul embarqués performants est activement maintenu et propose les innovations récentes, comme YOLOv4 qui est le nouvel état de l'art de cette famille.

Dans la section 3.4, nous réaliserons une étude expérimentale des réseaux de la famille de YOLO et nous conduirons une recherche détaillée du paramétrage optimal de ces réseaux afin de l'exploiter pour la localisation d'organes de plantes et l'estimation du nombre de feuilles. Cette contribution ainsi que le paramétrage et l'étude expérimentale des réseaux de la famille YOLO ont fait l'objet d'une publication [76].

2.3 Approche neuronale profonde pour la détection de structure de plantes

Cette section présente une méthode d'apprentissage profond conçue spécifiquement pour la caractérisation de plante et la localisation d'organes de plantes. En effet, la première approche présentée dans la section 2.2 est limitée par certaines contraintes des détecteurs d'objets, par exemple, le nombre réduit de feuilles pouvant être appris pour chaque espèce de plante. La méthode développée propose une architecture neuronale capable de prédire la *structure* de plantes, définie comme la localisation des organes de plantes et leurs relations. Cette formulation permet d'unifier la régression de plusieurs propriétés de plantes : la détection d'instances (plantes entières), la localisation d'organes (tige, feuilles) et le nombre de feuilles de chaque plante.

La première section 2.3.1 présente les hypothèses et la définition du modèle de structure de plante élaboré tandis que la seconde section 2.3.2 explore les méthodes d'apprentissage profond qui peuvent être exploitées pour détecter la structure. La section 2.3.3 présente SDNet, le détecteur de structure de plante proposé.

2.3.1 Hypothèses et modélisation de la structure de plante

Les plantes présentent naturellement une structure arborescente complexe et variée. Quelques hypothèses ont été réalisées afin de simplifier le problème. Dans le cadre du projet BIPBIP présenté en section 1.1, les plantes sont traitées à un stade précoce. À ce stade, l'arborescence des organes de la plante est limitée, il n'y a généralement que quelques organes visibles (les feuilles). Ces feuilles sont reliées directement à la tige principale et il est rare d'observer des tiges secondaires (reliées à d'autres tiges).

Ces considérations qualitatives permettent d'ébaucher une modélisation de la *structure* de la plante basée sur les hypothèses suivantes :

- Une plante possède un point d'intérêt singulier nommé *ancre* qui n'est présent qu'en un seul exemplaire par instance de plante. Il s'agit du point d'entrée de la tige principale de la plante dans le sol.

- Les autres organes de la plante sont visuellement identifiables par un unique point d'intérêt non ambigu. C'est le cas des feuilles de la plante qui sont identifiables par la position de l'extrémité de la feuille (la pointe). Ces points d'intérêt sont nommés *parties*.

La figure 2.11 illustre quelques exemples sur le maïs (ligne du haut) et le haricot (ligne du bas) avec les ancres en couleur (rose pour le haricot et vert pour le maïs) et les parties en blanc. Les images de gauche présentent des cas simples avec une ou deux feuilles et les

FIGURE 2.11 : Illustration de la structure de plants de maïs (première ligne) et de haricots (seconde ligne). L'ancre est matérialisée par un point vert pour le maïs et rouge pour le haricot. Les feuilles (parties) sont matérialisées par un point blanc relié par un segment blanc à l'ancre correspondante.

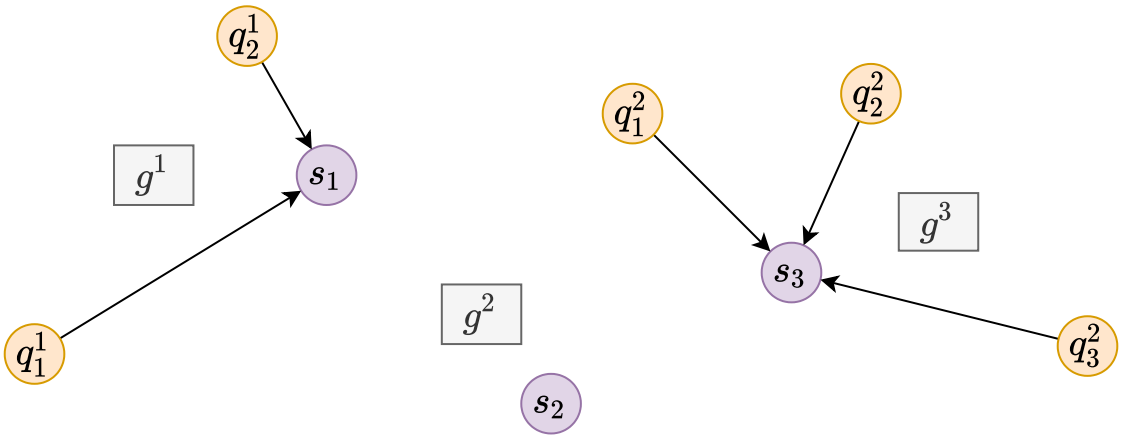


FIGURE 2.12 : Illustration de trois graphes en étoiles avec le nœud central (ancre) en violet et les sommets (parties) en jaune. g^1, g^2 et g^3 possèdent respectivement 2, 0 et 3 parties.

images de droite illustrent des cas plus compliqués où les plantes ont des tiges secondaires (haricot de droite) ou des feuilles aux formes plus complexes (maïs de droite).

Plus formellement, la définition de la structure de plante élaborée peut être décrite mathématiquement par un graphe en étoile, i.e. un graphe avec un nœud central et un nombre arbitraire de sommets d'ordre un. L'ancre correspond au nœud central et les parties aux sommets du graphe. Un graphe en étoile associe donc plusieurs parties à une unique ancre. La finalité de la méthode proposée est de détecter dans une image I_n un ensemble de graphes en étoile :

$$D_n = G = \left\{g^i\right\}_{i=1\ldots N_n}, \; g^i \in \mathcal{G}_{k_i}, \tag{2.9}$$

où N_n est le nombre de graphes détectés dans l'image n et $\mathcal{G}_{k_i}$ la famille de graphes en étoile avec k_i sommets. Un graphe g^i est composé d'un nœud central s^i (l'ancre) et de k_i sommets $Q^i = \{q^i_j\}_{j=1\ldots k_i}$, qui sont les parties de la structure :

$$g^i = s^i \cup \left\{q^i_j\right\}_{j=1\ldots k_i}. \tag{2.10}$$

Chaque nœud s^i (ancre ou partie) est défini par sa position spatiale (x_i, y_i) et sa classe c_i qui correspond à la variété de la plante pour l'ancre (maïs ou haricot par exemple) ou à la classe de la partie (une seule classe dans notre cas, la feuille) :

$$s^i = (x_i, y_i, c_i), \tag{2.11}$$

La figure 2.12 illustre un exemple d'ensemble de graphes en étoile ayant un nombre de sommets variable. Les ancres sont en violet et les parties en jaune.

L'intérêt d'utiliser les graphes en étoile pour modéliser la structure des plantes est qu'ils permettent d'une part de localiser les organes des plantes P_n dans une image I_n :

$$K_n = \left\{s^i\right\}_{i\in 1\ldots N_n} \cup \left\{q^i_j\right\}_{i\in 1\ldots N_n,\; j\in 1\ldots k_i}, \tag{2.12}$$

et, d'autre part, de connaître le nombre de parties k_i, i.e. le nombre de feuilles de chaque plante. De plus, le nombre de feuilles k_i n'est pas contraint contrairement à l'approche présentée dans la section 2.2 et il est en théorie illimité.

Néanmoins, à notre connaissance, il n'existe pas de réseau capable de prédire des graphes en étoile excepté ceux qui se limitent à des cas particuliers. La section 2.3.2 explore les méthodes qui s'approchent le plus de ce but, puis la section 2.3.3 présente une architecture neuronale capable de régresser des graphes en étoile.

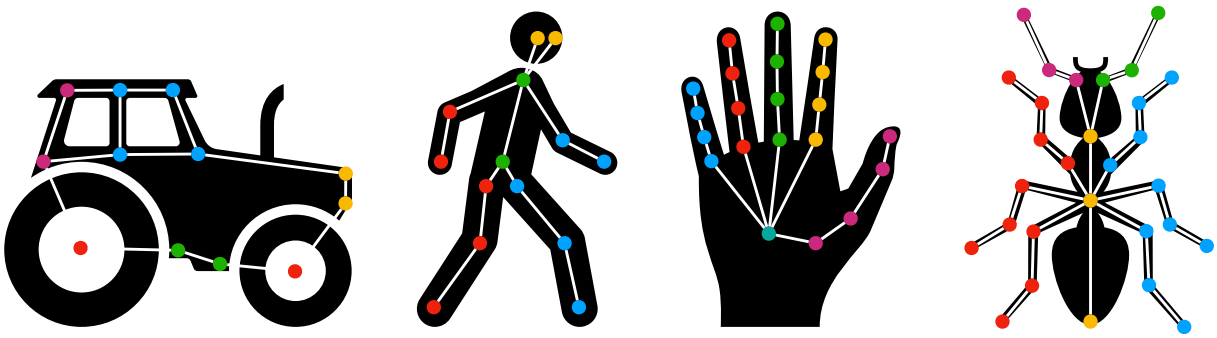


FIGURE 2.13 : Illustrations de différentes poses d'objets. De gauche à droite : pose de véhicule, pose humaine, pose de la main et pose animale. Une pose est définie par un graphe où les sommets représentent les points d'intérêt (points de couleur sur la figure) de la pose (articulations par exemple) et les arêtes (segments blancs) représentent les connections entre les points d'intérêt.

2.3.2 Méthodes d'estimation de pose d'objets

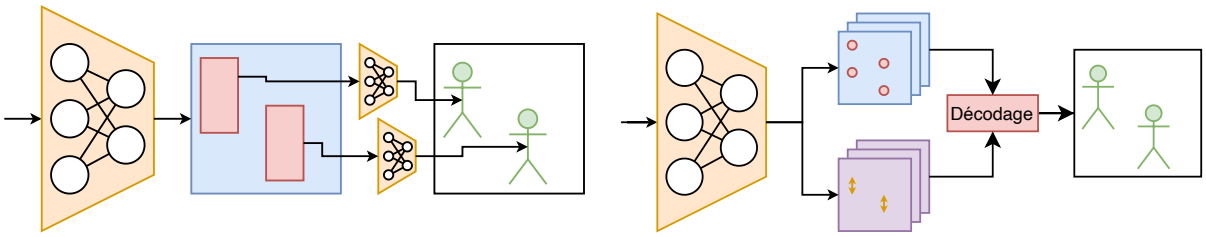
Les méthodes s'approchant le plus de la détection de graphes en étoile sont les détecteurs de pose d'objets car ils permettent de prédire des graphes, i.e. un ensemble de structures formées d'objets (les *nœuds* ou *sommets*) qui sont en relation (les *arêtes*). En particulier, la *pose* d'un objet est définie par un ensemble de points d'intérêt (les sommets du graphe) et de connexions entre eux (les arêtes). La figure 2.13 illustre quelques exemples de poses. Les points d'intérêt identifient généralement des parties spécifiques de l'objet comme les articulations pour la pose humaine, la pose de main ou la pose animale. Les liens identifient les connections physiques entre les points d'intérêt, telles que les membres entre deux articulations.

Les architectures neuronales utilisées pour la détection de pose sont nombreuses. Comme pour les réseaux de détection d'objets, il y a deux catégories de détecteurs de pose, illustrés dans la figure 2.14, qui se différencient par le compromis vitesse-précision qu'elles permettent d'atteindre :

- **L'approche descendante** (*top-down* dans la littérature), qui est analogue à la catégorie de détecteurs d'objets à deux étages. Un premier réseau de neurones apprend à détecter les instances d'objets (par exemple les boîtes englobantes). Ensuite, un second réseau de neurones régresse la pose de chaque objet. Cette approche est illustrée sur la figure 2.14a.

- **L'approche ascendante** (*bottom-up* dans la littérature), qui est analogue à la catégorie des détecteurs d'objets à un étage. Un seul réseau de neurones est chargé de régresser les points d'intérêt et les liens de tous les objets. Ensuite, un algorithme de *décodage* reconstruit la pose individuelle des objets à partir de ces informations. Cette approche est illustrée sur la figure 2.14b.

Pour les mêmes raisons relatives au compromis vitesse-précision des architectures à un



(a) Approche descendante (b) Approche ascendante

FIGURE 2.14 : Illustration des approches descendantes et ascendantes pour la détection de pose. L'approche descendante emploie un premier réseau de neurones pour détecter les objets puis un second pour régresser la pose individuelle de chaque objet. L'approche ascendante emploie un seul réseau pour régresser tous les points d'intérêt (points rouges) et les liens (flèches oranges) puis reconstruit les poses individuelles via un algorithme dit de décodage.

et deux étages faites en section 2.2, les approches descendantes sont plus précises mais leur vitesse d'exécution est plus lente et, à l'inverse, les approches ascendantes sont plus rapides mais moins précises. Comme pour le choix de la famille de réseaux YOLO motivé dans la section 2.2.3, ce travail s'est concentré sur les approches ascendantes car elles sont plus adaptées à une exécution en temps réel.

Le réseau de détection de pose top-down le plus courant est Mask R-CNN [55], utilisé pour la détection de la pose humaine. Les réseaux bottom-up sont plus nombreux. Notamment, il y a le réseau PifPaf [73], OpenPose [21, 22], le réseau Hourglass [8], la méthode "Associative Embeddings" [102], CenterNet [142] et d'autres [137]. La majorité de ces architectures sont utilisées pour la détection de pose humaine mais elles ne s'y limitent pas : la flexibilité de l'apprentissage profond permet d'employer la même architecture neuronale dans des situations très différentes avec très peu d'adaptation. Par exemple, le détecteur de pose humaine [22] est utilisé dans [115] pour détecter la pose d'abeilles. Dans [92], c'est l'architecture Hourglass [8] qui est exploitée pour détecter la pose de véhicules, et dans [60] il est employé pour la détection de la pose de la main. L'article sur CenterNet [142] démontre son utilisation pour la détection de la pose en 3D de véhicules.

Les réseaux de détection ascendante de pose d'objets se ressemblent beaucoup sur le principe de fonctionnement. Le réseau est chargé d'apprendre deux types de données qui sont ensuite utilisées par le décodeur :

- **Les cartes de chaleur** (*heatmaps*), utilisées pour prédire la position des points d'intérêt. Comme leur nom l'indique, les cartes de chaleur sont des matrices avec une valeur haute (usuellement 1) aux localisations où il y a une forte probabilité de présence d'un point d'intérêt et une valeur basse (usuellement 0) aux autres positions. Par exemple, pour un détecteur de pose humaine, il y a une carte de chaleur pour chaque articulation : coude, genou, poignet, etc.

- **Des indices d'association** ou de regroupement (*embeddings*), qui permettent

d'associer plusieurs points d'intérêt entre eux. Pour reprendre l'exemple du détecteur de pose humaine, un indice d'association permet d'indiquer à quel point d'intérêt d'épaule correspond un point d'intérêt de coude donné.

Le rôle du décodeur est donc de regrouper les points d'intérêt qui correspondent à la même entité grâce aux indices de regroupement. En général, les détecteurs de pose ne sont pas prévus pour prédire des graphes arbitraires puisque cela n'a pas vraiment d'intérêt pour des poses "fixes" comme la pose humaine ou la pose animale habituellement visées. En effet, ces objets possédant une structure fixe, le nombre de points d'intérêt est toujours le même par objet et les liens associent toujours les mêmes points entre eux. Par exemple, un humain ne possède qu'un bras droit et celui-ci est toujours relié à un seul point : l'épaule droite. Un autre exemple est la pose de la main qui ne possède que cinq doigts, il est donc inutile pour un détecteur d'offrir la possibilité de détecter six doigts ou plus.

Pour résumer, les détecteurs de pose d'objets traditionnels poursuivent un but proche de la détection de structure de plante puisqu'ils permettent de prédire certaines classes de graphes. En revanche, ils ne sont pas assez flexibles pour autoriser la prédiction de graphes étoilés ayant un nombre arbitraire de sommets. Néanmoins, il semble possible d'améliorer leur fonctionnement et de concevoir un détecteur capable de prédire de telles structures. La section suivante 2.3.3 présente SDNet, une architecture neuronale originale capable de prédire des graphes étoilés.

2.3.3 SDNet : une architecture neuronale de détection de graphes étoilés

La conception de SDNet (*Structure Detection Network*) est basée sur une architecture ascendante afin de pouvoir l'employer pour une application en temps réel. Un unique réseau de neurones est donc utilisé pour prédire l'ensemble des cartes de chaleur et des indices d'association, ensuite analysés par un décodeur reconstruisant les structures de plantes. La figure 2.15 illustre ce fonctionnement sur un exemple d'image. Le code source de SDNet est publié en open source[7].

Les différentes architectures de détection de pose ascendante présentées dans la section 2.3.2 se distinguent principalement par la formulation des indices de regroupement et par l'algorithme de décodage. La régression des cartes de chaleur fonctionne généralement de la même manière pour toutes les architectures. Par exemple, le réseau PifPaf [73] utilise un champ de vecteurs qui joint deux points d'intérêt. Lors du décodage, l'algorithme démarre de la position d'un point puis "suit" le flux du champ de vecteurs jusqu'à un autre point. Les points d'intérêt sont donc agglomérés itérativement et à la fin du processus chaque groupe forme un objet. OpenPose [21, 22] utilise une définition similaire pour les indices de regroupement mais implémente un algorithme de décodage différent. La structure de graphe

[7] `github.com/laclouis5/StructureDetector`

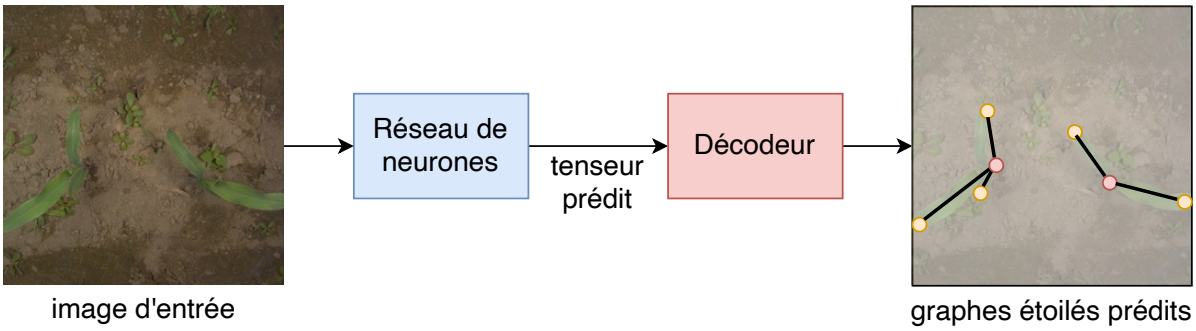


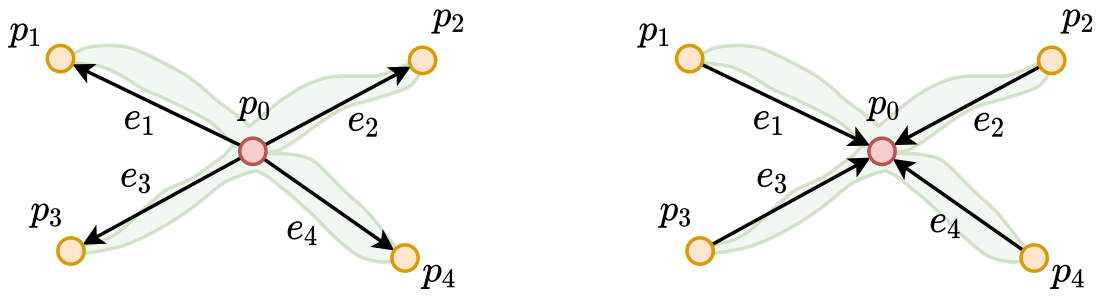
FIGURE 2.15 : Schéma général du fonctionnement de SDNet. Basé sur une approche ascendante, un seul réseau de neurones est employé et un algorithme de décodage est chargé de prédire les graphes en étoile (structures de plantes).

y est mieux explicitée et un algorithme d'appariement de graphe bipartite est développé. L'article Newell, Huang et Dang [102] emploie des indices associatifs assignant une valeur identique aux points d'intérêt appartenant au même objet et une valeur éloignée pour les points n'appartenant pas au même objet. Le décodeur réalise ensuite une association des points ayant un indice de regroupement de valeur similaire. CenterNet [142] utilise des vecteurs appris seulement à la position d'un point d'intérêt spécifique, appelé ancre, unique pour chaque objet. Ces vecteurs "pointent" vers les autres points d'intérêt de l'objet. Le décodeur détecte donc d'abord les ancres puis y associe les points d'intérêt en "suivant" les vecteurs.

L'architecture de SDNet est basée sur celle de CenterNet [142]. Cette dernière est à la fois simple et facilement interprétable. Ainsi, contrairement à PifPaf et OpenPose, elle ne nécessite pas d'algorithme d'appariement de graphe ou de méthode de "suivi" du flux d'un champ de vecteurs, et, contrairement à [102], les indices de regroupement ne sont pas arbitraires[8] et ont un sens physique qui représente concrètement le lien entre deux points d'intérêt. La figure 2.16a illustre de manière simplifiée la définition des indices de regroupement utilisée par CenterNet pour détecter des objets à cinq points d'intérêt au maximum. Pour une ancre p_0 (point d'intérêt en rouge), quatre vecteurs e_1 à e_4 sont appris et pointent vers les quatre autres points d'intérêt de l'objet p_1 à p_4 (en orange). La limite de cette approche est qu'elle ne permet pas de prédire un nombre arbitraire de parties, seulement quatre au maximum dans l'exemple présenté.

La stratégie proposée pour surmonter ce problème et illustrée dans la figure 2.16b est de ne pas centraliser l'apprentissage des indices de regroupement dans l'ancre mais de le distribuer sur les parties. Chaque partie p_i apprend un unique vecteur e_i qui pointe vers l'ancre correspondante. Le décodeur est ensuite chargé de regrouper pour chaque ancre les parties venant s'y "connecter". Cette approche permet de rendre variable le nombre de parties qui peuvent être associées à une ancre donnée et donc de prédire des graphes en

[8]Dans [102] les indices ne sont pas appris directement mais via des contraintes de distance entre indices (valeur proche pour les points d'un même objet, sinon valeur éloignée). Il n'y a pas de contrainte sur la valeur, le réseau apprend donc une valeur arbitraire qui n'a pas de signification particulière.



(a) Indices de regroupement centralisés (CenterNet [142])

(b) Indices de regroupement décentralisés (SDNet [77])

FIGURE 2.16 : Illustration de deux définitions d'indices de regroupement par des vecteurs d'association pour une plante à quatre feuilles. Les ancres sont en rouge, les parties en orange et les vecteurs d'indice en noir (flèches).

étoiles ayant un nombre variable de sommets.

Les sous-sections suivantes détaillent l'implémentation de cette stratégie dans une architecture neuronale dont les différents composants sont :

- **Le réseau de neurones profond** qui est chargé d'apprendre à régresser les données brutes (tenseurs) : les cartes de chaleur servant à détecter les points d'intérêt, les indices de regroupement utilisés pour associer les points d'intérêts, et les vecteurs de compensation de la décimation dont l'utilité est détaillée plus loin.

- **L'encodeur** qui est un algorithme utilisé lors de l'apprentissage pour générer les vérités de terrain nécessaires à l'entraînement via la fonction de coût. Sa fonction est de convertir les annotations de données en tenseurs de référence.

- **La fonction de coût** qui évalue la performance du réseau et lui permet d'apprendre à régresser les cartes de chaleur et les indices de regroupement.

- **Le décodeur** qui est chargé de reconstituer les graphes étoilés à partir des cartes de chaleur et des indices de regroupement.

2.3.3.1 Réseau de neurones profond

Le réseau de neurones est chargé de prédire les cartes de chaleur, les indices de regroupement et les vecteurs de compensation de la décimation. Afin d'obtenir des cartes de chaleur d'une définition suffisante (i.e. capable de distinguer des points d'intérêt très proches), une architecture neuronale couramment employée est celle de l'encodeur-décodeur (à ne pas confondre avec les algorithmes d'encodage et de décodage des graphes en étoile), illustrée dans la figure 2.17 :

- La colonne du réseau de neurones (l'encodeur) est constituée de couches convolutives suivies de couches de réduction de la dimension spatiale (*max-pooling* par exemple) et

de non-linéarité (*relu* par exemple) réduisant progressivement la dimension spatiale au profit de la dimension sémantique (nombre de canaux). Cela permet de générer des descripteurs de haut niveau, mais qui sont peu localisés (perte d'information spatiale due à la réduction de dimension spatiale).

- La nuque du réseau de neurones (le décodeur) est constituée de couches dites déconvolutives[9] qui permettent de retrouver progressivement l'extension spatiale perdue. Des connexions latérales permettent d'acheminer l'information très localisée des premières couches de l'encodeur dans les descripteurs de haut-niveau peu localisés. Il en résulte un ensemble de descripteurs de haut-niveau bien localisés.

Cette architecture est, par exemple, présente dans les réseaux Hourglass [103] et UNet [116]. L'intérêt du décodeur est d'avoir une définition spatiale plus élevée des descripteurs. Ainsi, un encodeur de 4 couches (chaque couche étant composée d'une convolution suivie d'une réduction de dimension et d'une non-linéarité) divise par 16 la définition par rapport à celle de l'image d'entrée, ce qui représente une perte d'information spatiale conséquente. Ajouter un décodeur de 2 couches augmente la définition du descripteur de 4, soit une réduction totale de la définition de seulement 4. On nomme le ratio R entre la taille de l'image d'entrée et la taille du descripteur de sortie le *facteur d'échelle* (*output stride* dans la littérature), qui est en pratique un *facteur de réduction* (supérieur à 1) car produire un descripteur de haute définition spatiale est très coûteux en temps de calcul[10]. Afin de compenser la perte de définition causée par la décimation de facteur R, il est proposé, comme pour CenterNet [142] et PifPaf [73], d'estimer un *champ vectoriel de compensation de décimation*.

Les choix d'implémentation pour la colonne, la nuque et la tête du réseau sont essentiellement guidés par des considérations de performances vis-à-vis du compromis vitesse-précision ainsi que de rapidité de convergence de l'entraînement. Un test exhaustif de toutes les configurations de ces trois éléments est coûteux et chronophage car cela requiert une recherche exhaustive (*grid search* en anglais) nécessitant un entraînement complet par itération. Le choix s'est porté sur une architecture simple, reconnue comme performante dans la littérature, appropriée à l'utilisation en temps réel et à la taille relativement faible du jeu de données d'entraînement. Ainsi, la colonne choisie est un réseau à apprentissage résiduel [56] nommé *resnet34*, la nuque est un FPN [84] de 3 connections latérales à 128 canaux de descripteurs, et la tête du réseau est une simple couche convolutive. Une normalisation de batch [62] et une fonction d'activation non-linéaire de type *relu* sont utilisées après chaque couche convolutive, excepté dans la couche de la tête. Le facteur de réduction R de cette architecture est de 4, e.g. une image d'entrée de taille 512×512

[9] Les couches déconvolutives, aussi appelées convolutions transposées, sont constituées d'une couche de sur-échantillonnage (augmentation de définition spatiale) suivie d'une couche convolutive.

[10] Les réseaux convolutifs utilisent des convolutions qui sont des opérateurs ayant une complexité calculatoire en $\mathcal{O}(W \times H)$ où (W, H) est la définition du descripteur en entrée. Plus le descripteur a une haute définition, plus ce calcul est coûteux.

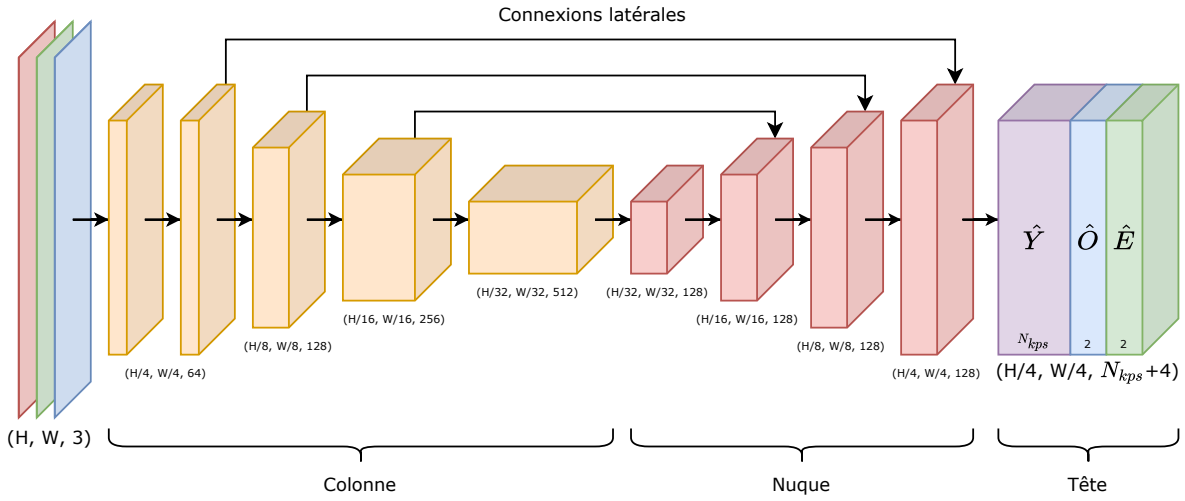


FIGURE 2.17 : Schéma du réseau de neurones de SDNet. La colonne est présentée en jaune, la nuque en rouge avec les connexions latérales représentées par des flèches et la tête du réseau par les tenseurs violet (cartes de chaleur des N_{kps} points d'intérêt, ancres et parties), bleu (champ vectoriel de compensation de la décimation) et vert (champ vectoriel des indices de regroupement). Les tailles de tenseur sont indiquées au format (hauteur, largeur, canal).

produira un tenseur de sortie ayant une extension spatiale de 128 × 128. L'ensemble de cette architecture est schématisé dans la figure 2.17.

Plus formellement, SDNet prend en entrée une image $I \in \mathbb{R}^{H \times W \times 3}$ où H est la hauteur et W la largeur de l'image. Il a pour rôle de prédire trois types de données :

- Un ensemble de cartes de chaleur $\hat{Y} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times N_{kps}}$ où N_{kps} est le nombre de types de points d'intérêt (ancres et parties). Par exemple, pour détecter la structure de deux types de plantes (maïs et haricot) et d'une partie commune (les feuilles), il y a $N_{anc} = 2$ ancres et $N_{par} = 1$ partie.

- Un champ vectoriel d'indices de regroupement $\hat{E} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 2}$. Pour chaque partie, l'indice de regroupement "pointe" vers l'ancre qui correspond à l'objet dont elle fait partie (voir figure 2.16b).

- Un champ vectoriel de compensation de décimation $\hat{O} \in \mathbb{R}^{\frac{H}{R} \times \frac{W}{R} \times 2}$ qui permet d'obtenir les positions non décimées des points d'intérêt à partir de positions décimées.

2.3.3.2 Encodage des annotations

L'encodeur désigne l'algorithme transformant les données d'entraînement dans un format permettant de superviser l'apprentissage de $\hat{Y}$, $\hat{E}$, et $\hat{O}$. Les données d'entraînement (annotations) correspondent aux vérités de terrain des graphes en étoile G de chaque image. Par exemple, dans le cas de la détection de structure de plante, l'encodeur génère les cartes de chaleur Y correspondant aux points d'intérêt de tiges et de feuilles et crée les champs

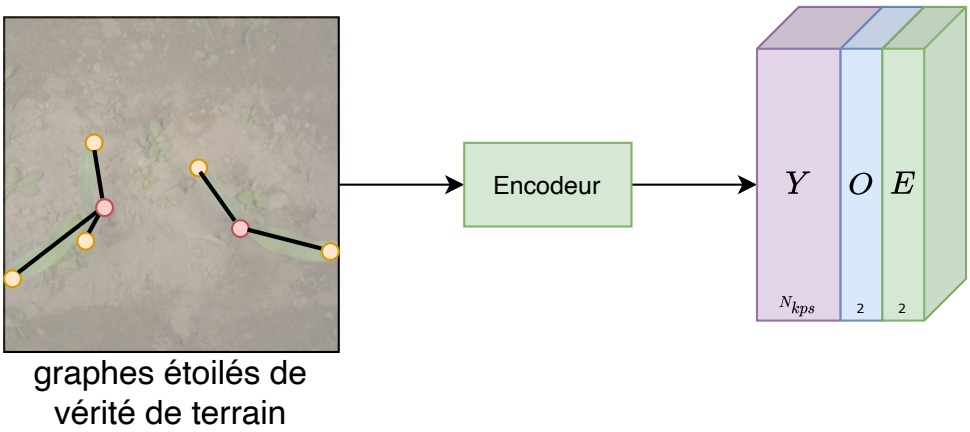


FIGURE 2.18 : Illustration du processus d'encodage des annotations de SDNet. L'encodeur (en vert) transforme les annotations de graphes étoilés de l'image d'entrée en tenseurs d'entraînement Y, E et O.

vectoriels d'indices de regroupement E et de compensation de décimation O. La figure 2.18 illustre ce processus d'encodage.

Cartes de chaleur L'apprentissage des cartes de chaleur est supervisé à toutes les positions spatiales, i.e. le réseau apprend sur des exemples positifs (là où il y a un point d'intérêt) et négatifs (là où il n'y a aucun point). Chacune des N_{kps} cartes de chaleur correspond à un type de point d'intérêt (ancre ou partie). La carte de chaleur du type de point $c \in 1 \dots N_{kps}$ est à une valeur haute (proche de 1) aux positions où il y a une forte probabilité de présence d'un point d'intérêt de type c et est à une valeur basse (proche de 0) dans le cas contraire. Pour chaque point d'intérêt de position $\tilde{p} = \lfloor \frac{p}{R} \rfloor = (\tilde{x}, \tilde{y})$ (position décimée) et de classe c, la valeur de la carte de chaleur $Y_{c,\tilde{p}}$ à la position $(x, y) \in \mathbb{R}^2$ est définie par :

$$Y_{c,\tilde{p}}(x, y) = \exp\left(-\frac{(x-\tilde{x})^2 + (y-\tilde{y})^2}{2{\sigma_e}^2}\right). \tag{2.13}$$

La carte de chaleur Y_c de l'ensemble des points d'intérêt de type c est obtenue par réduction des cartes de chaleurs individuelles via l'opérateur `max`, comme pour OpenPose [21] :

$$Y_c(x, y) = \max_{\tilde{p} \in K_c} Y_{c,\tilde{p}}(x, y), \tag{2.14}$$

où K_c désigne l'ensemble des points d'intérêt de type c. De la même manière que pour CenterNet [142], le filtrage gaussien permet de réduire le coût aux positions proches d'un point d'intérêt afin d'apprendre plus rapidement les cartes de chaleur. La valeur de σ_e module la largeur des courbes gaussiennes. Une recherche exhaustive sur ce paramètre est présentée dans la section 3.3.

Vecteurs de compensation de décimation Détecter des points d'intérêt via une carte de chaleur implique une perte d'information due à la décimation nécessaire pour créer les

cartes de chaleur. En effet, l'image I est usuellement en haute définition (par exemple 2048 par 1536) et un point d'intérêt de coordonnée p exprimée dans le référentiel $\mathcal{R}$ de l'image est généralement finement localisé (2048 valeurs possibles sur l'axe horizontal). Cependant, malgré l'utilisation d'un décodeur pour limiter la chute du facteur de réduction du réseau, la définition des cartes de chaleur est faible (par exemple, pour une image en entrée[11] du réseau de 512×512, la définition des cartes de chaleur est de 128×128 avec un facteur de réduction de $R = 4$). Cela implique que la position $\tilde{p}$ de p exprimée dans le référentiel des cartes de chaleur soit sous-résolue (seulement 128 valeurs possibles sur l'axe horizontal). On connaît exactement cette perte de précision causée par la décimation puisqu'il s'agit par définition de la partie fractionnaire de $\frac{p}{R}$. De plus, elle peut être apprise par le réseau : il suffit de l'apprendre à la localisation du point d'intérêt exprimée dans le référentiel des cartes de chaleur. Cette approche est, par exemple, employée dans les réseaux PifPaf [73] et CenterNet [142]. Pour chaque point d'intérêt de position p, la valeur du champ de vecteurs de compensation O est :

$$O_{\tilde{p}} = \frac{p}{R} - \left\lfloor \frac{p}{R} \right\rfloor . \tag{2.15}$$

Les vecteurs de compensation de la décimation ne sont supervisés qu'aux localisations de points d'intérêt contrairement aux cartes de chaleur qui sont supervisées à toutes les positions.

Indices de regroupement Les indices de regroupement consistent en des vecteurs appris uniquement aux localisations des points d'intérêt de parties d'objets (par exemple, les feuilles pour la détection de structure de plantes) qui "pointent" vers l'ancre correspondant à l'objet (la tige pour la détection de structure de plantes) auquel appartient la partie. L'apprentissage des indices de regroupement n'est donc supervisé qu'aux localisations des parties contrairement aux cartes de chaleur. Par définition, pour toute partie de localisation $e = (e_x, e_y)$ correspondant à une ancre de localisation a, la valeur du champ de vecteurs d'indices E s'exprime :

$$E_{\tilde{e}} = e - a, \ \tilde{e} = \left\lfloor \frac{e}{R} \right\rfloor , \tag{2.16}$$

où $\tilde{e}$ est la localisation décimée de e dans le référentiel des cartes de chaleur.

Synthèse Pour résumer, le processus d'encodage consiste en la régression de plusieurs tenseurs :

- Les cartes de chaleur servent à localiser tous les points d'intérêt, ancres et parties. Un filtrage gaussien est utilisé pour améliorer l'apprentissage qui est supervisé à toutes les positions spatiales.

[11] Pour simplifier, l'étape de sous-échantillonnage de I pour l'adapter à la taille d'entrée du réseau n'est pas présentée car en pratique elle n'a pas d'impact sur la définition de la localisation p.

- Les vecteurs de compensation de la décimation sont appris et supervisés uniquement aux positions des points d'intérêt, ancres et parties. Ils servent à récupérer la précision perdue par la décimation introduite par les cartes de chaleur.

- Les vecteurs d'indices de regroupement sont appris (et donc supervisés) aux positions de parties. Ce sont des vecteurs qui pointent vers la position estimée de l'ancre correspondant à la partie.

2.3.3.3 La fonction de coût

L'apprentissage d'un réseau de neurones est basé sur l'optimisation d'une fonction de coût caractérisant la précision de la prédiction réalisée par le réseau vis-à-vis d'une vérité de terrain (annotation). C'est une fonction qui retourne un scalaire positif ou nul : une valeur proche de zéro indiquant un apprentissage presque parfait tandis qu'une valeur élevée signalant qu'une amélioration est possible. C'est une fonction différentiable dont la dérivée pilote la manière dont il faut mettre à jour le réseau de neurones pour qu'il soit plus performant. Sa définition est donc cruciale pour l'apprentissage. Des détails complémentaires sur la théorie des réseaux de neurones sont fournis en annexe A.

Pour SDNet, il y a trois objectifs à optimiser, soit trois fonctions de coût : une fonction pour les cartes de chaleur (L_y), une autre fonction pour les vecteurs de compensation (L_o) et une troisième fonction pour les indices de regroupement (L_e). La fonction de coût global de SDNet est une somme pondérée des trois fonctions de coût :

$$L = \lambda_y L_y + \lambda_o L_o + \lambda_e L_e \tag{2.17}$$

Pour les cartes de chaleur, la norme L2 est souvent employée [103, 102, 137, 142, 21] et plus rarement un FocalLoss [85] est utilisé [142]. La norme L2 est choisie pour SDNet en raison de sa simplicité :

$$L_y = \frac{1}{N_{cel}} \sum_{x,y,c} \left(Y_{xyc} - \mathrm{sig}(\hat{Y}_{xyc}) \right)^2 , \tag{2.18}$$

où $N_{cel} = \frac{WHC}{R^2}$ est le nombre de cellules de la carte de chaleur et sig la fonction sigmoïde (utilisée pour "comprimer" les valeurs des cartes de chaleur dans l'intervalle $0 \ldots 1$). Pour les prédictions relatives à des vecteurs de l'espace, comme les vecteurs de compensation et les indices de regroupement, la norme L1 (ou une variante) est préférée [142, 73] bien que la norme L2 puisse aussi être utilisée [21, 22][12]. Pour SDNet, la norme L1 est utilisée pour

[12] Il ressort de la littérature (sans que cela soit démontré, comme beaucoup de décisions algorithmiques dans le domaine de l'apprentissage profond) que la norme L2 est jugée plus adaptée pour les cartes de chaleur et la norme L1 est préférée pour les vecteurs de l'espace.

les vecteurs de compensation de décimation :

$$L_o = \frac{1}{N_p} \sum_{\tilde{p}} \left| O_{\tilde{p}} - \hat{O}_{\tilde{p}} \right|, \tag{2.19}$$

et les indices de regroupement :

$$L_e = \frac{1}{N_e} \sum_{\tilde{e}} \left| E_{\tilde{e}} - \hat{E}_{\tilde{e}} \right|, \tag{2.20}$$

où N_p et N_e désignent respectivement le nombre de points d'intérêt et le nombre de parties d'objet de la vérité de terrain, et $|\cdot|$ désigne la valeur absolue. Les facteurs de pondération λ_y, λ_o et λ_e sont des constantes réelles strictement positives choisies empiriquement afin que le processus d'apprentissage converge.

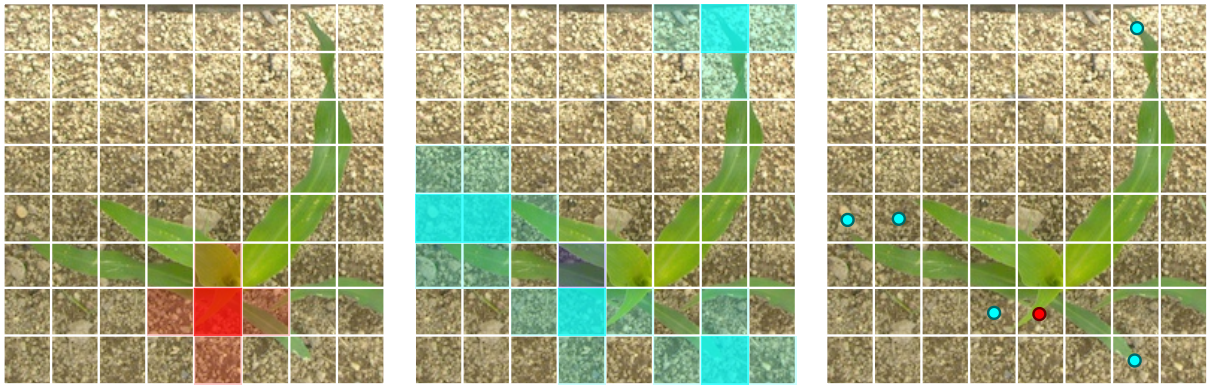
2.3.3.4 Décodage

Le décodage désigne l'algorithme qui, étant donné des tenseurs de sortie du réseau $\hat{Y}$, $\hat{O}$ et $\hat{E}$, produit un ensemble $\hat{G}$ de graphes étoilés représentant la structure des objets présents dans l'image d'entrée. Il est réalisé en trois étapes :

- L'extraction des points d'intérêt des ancres et parties.
- La compensation de décimation des points d'intérêt.
- L'association des parties aux ancres afin de produire l'ensemble des graphes étoilés.

Extraction des ancres et parties La première étape consiste en l'extraction des points d'intérêt d'ancres et de parties d'objets depuis les cartes de chaleur prédites $\hat{Y}$. Les cartes de chaleur ont une valeur haute (proche de 1) aux endroits où la probabilité de présence d'un point d'intérêt est forte et une valeur faible (proche de 0) aux endroits sans point d'intérêt. Les figures 2.19a et 2.19b illustrent respectivement les cartes de chaleur d'ancre et de partie sur un exemple de maïs. Une valeur haute correspond à une forte opacité et une valeur basse à une faible opacité (transparence). Détecter la position des points d'intérêt consiste à détecter les cellules de la carte de chaleur de plus forte intensité. Afin de supprimer les doublons potentiels, une opération de détection des maxima locaux est d'abord réalisée puisque plusieurs cellules de forte intensité peuvent être adjacentes. Un max-pooling de taille 5×5 est appliqué indépendamment sur chaque carte de chaleur :

$$\hat{Y}^f = \text{maxpool}(\hat{Y}^s), \tag{2.21}$$



(a) (b) (c)

FIGURE 2.19 : Extraction des ancres (en rouge) et parties (en bleu) illustré sur une image de maïs à partir des cartes de chaleur $\hat{Y}$. L'opacité des cartes de chaleur des ancres (a) et des parties (b) correspond au score de confiance sur la présence d'un point d'intérêt. La sous-figure (c) illustre les ancres $\hat{S}$ et parties $\hat{Q}$ prédites par filtrage des cartes de chaleur avec le seuil sur la confiance t_c. Les positions sont décimées (les points d'intérêts sont au centre des cellules de la carte de chaleur). Le facteur R est volontairement élevé pour clarifier la figure.

où $\hat{Y}^s = \text{sig}(\hat{Y})$. Ensuite, seuls les pics locaux sont gardés, i.e. les cellules de la carte de chaleur qui correspondent au maximum d'intensité sur le voisinage de taille 5×5 :

$$\hat{P}^f = \left\{(x, y, c, l),\ \hat{Y}^f_{xyc} = \hat{Y}^s_{xyc}\right\}_{x \in 1 \ldots \frac{W}{R},\ y \in 1 \ldots \frac{H}{R},\ c \in 1 \ldots N_{kps}}, \tag{2.22}$$

où $l \in 0 \ldots 1$ est la valeur d'intensité $\hat{Y}^s_{xyc}$ de la cellule de la carte de chaleur c à la position (x, y) (i.e. le score de confiance). Un dernier filtrage supprime les points d'intérêt dont le score de confiance est inférieur à un seuil t_c :

$$\hat{P} = \{p,\ p_l > t_c\}_{p \in \hat{P}^f}, \tag{2.23}$$

où p_l est le score de confiance de p. Il en résulte un ensemble de points d'intérêt $\hat{P}$ qui peut être séparé en un ensemble de points d'ancres $\hat{S}$ et un ensemble de points de parties $\hat{Q}$. À cette étape, les coordonnées des points d'intérêt sont décimées et donc peu précises puisqu'elles correspondent aux coordonnées entières des positions des cellules dans les cartes de chaleur. La figure 2.19c illustre les ancres $\hat{S}$ (en rouge) et parties $\hat{Q}$ (en bleu) prédites à partir des cartes de chaleur simplifiées présentées dans la figure 2.19.

Compensation de la décimation Les ancres et parties régressées à ce point sont exprimées en coordonnées entières dans les intervalles $[1, \frac{W}{R}]$ et $[1, \frac{H}{R}]$ puisque la valeur de position correspond à la coordonnée du pic local dans la carte de chaleur. La partie fractionnaire manquante est apprise via le champ vectoriel de compensation de la décimation $\hat{O}$ à la localisation de chaque point d'intérêt. Pour compenser la perte de précision due à

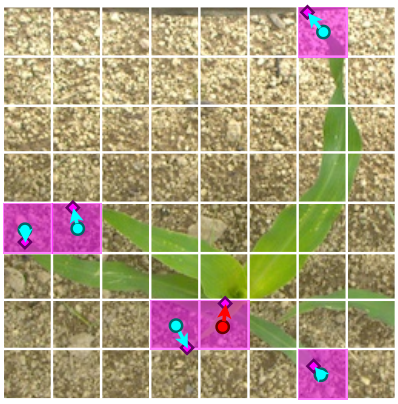


FIGURE 2.20 : Illustration de la compensation de décimation sur une image de maïs. Les vecteurs de compensation (flèches) régressés aux positions décimées des points d'intérêt (cellules en rose) sont ajoutés aux coordonnées des ancres décimées $\hat{S}$ (points rouges) et parties décimées $\hat{Q}$ (points bleus) pour obtenir les ancres $\hat{S}'$ et parties $\hat{Q}'$ compensées (losanges roses). Voir la figure 2.19 pour une illustration de la prédiction des ancres et parties décimées.

la décimation, il suffit donc d'ajouter la partie fractionnaire prédite à la position décimée des ancres et parties dans le champ vectoriel $\hat{O}$ (cellules en rose sur la figure 2.20) :

$$\begin{aligned} \hat{S}' &= \left\{ \left(s_x + \hat{O}^x_{s_x s_y}, s_y + \hat{O}^y_{s_x s_y}, c, l \right), \ s \in \hat{S} \right\} \\ \hat{Q}' &= \left\{ \left(q_x + \hat{O}^x_{q_x q_y}, q_y + \hat{O}^y_{q_x q_y}, c, l \right), \ q \in \hat{Q} \right\}, \end{aligned} \tag{2.24}$$

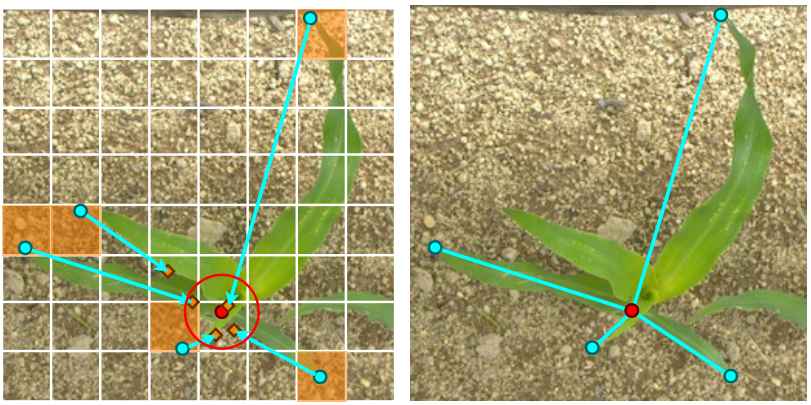
où (s_x, s_y) est la coordonnée entière (décimée) de l'ancre s, c est la classe de l'ancre, l est le score de confiance et $(\hat{O}^x_{s_x s_y}, \hat{O}^y_{s_x s_y})$ est le vecteur de compensation de s (mêmes notations pour les parties $\hat{Q}$). La figure 2.20 illustre les ancres décimées prédites $\hat{S}$ par des points rouges, les parties décimées $\hat{Q}$ par des points bleus, les vecteurs de compensation de la décimation par des flèches et les ancres et parties compensées par des losanges roses.

Association des parties aux ancres La dernière étape du décodage consiste en l'association des parties $\hat{Q}'$ aux ancres $\hat{S}'$ via les indices de regroupement prédits $\hat{E}$. Les indices de regroupement sont appris aux positions (décimées) des parties d'objets uniquement. Ils consistent en des vecteurs qui pointent vers l'ancre correspondant à l'objet auquel la partie appartient. Chaque partie est ensuite associée à l'ancre prédite la plus proche, s'il y en a une. Dans un premier temps, les positions des ancres prédites de chaque partie, appelées *ancres projetées*, sont calculées :

$$\hat{S}^* = \left\{ \left(q'_x + \hat{E}^x_{q_x q_y}, q'_y + \hat{E}^y_{q_x q_y}, c, l \right), \ q' \in \hat{Q}' \right\}, \tag{2.25}$$

où q' est le point d'intérêt compensé correspondant à q et $(\hat{E}^x_{q_x q_y}, \hat{E}^y_{q_x q_y})$ est le vecteur d'indice de regroupement de q.

Les indices de regroupement de parties sont illustrés par des flèches bleues sur la figure



(a) (b)

FIGURE 2.21 : Illustration de l'association des parties $\hat{Q}'$ aux ancres $\hat{Q}'$ sur une image de maïs. La figure a illustre les vecteurs d'indice de regroupement (flèches bleues) extraits aux positions des parties (cellules en orange), les ancres projetées $\hat{S}^*$ (losanges oranges) et le seuil de distance t_d (cercle rouge). La sous-figure b illustre le graphe étoilé final prédit.

2.21a et les ancres projetées par des losanges oranges. Dans un second temps, chaque ancre projetée (et donc chaque partie) est associée à l'ancre prédite la plus proche, i.e. pour chaque ancre $\hat{s}^i \in \hat{S}'$ prédite les parties d'objet associées $\hat{Q}_i$ sont :

$$\hat{Q}_i = \left\{ \hat{q}' \in \hat{Q}',\ \underset{\hat{s} \in \hat{S}'}{\operatorname{argmin}} \|\hat{s} - \hat{s}^*\| = \hat{s}^i \right\}, \tag{2.26}$$

où $\hat{s}^*$ est l'ancre projetée de la partie $\hat{q}'$ et $\|\cdot\|$ la distance Euclidienne. Une ancre projetée pouvant être relativement éloignée de l'ancre la plus proche. Un dernier filtrage supprime les parties d'une ancre dont l'ancre prédite est à une distance supérieure à un seuil t_d. Pour une ancre $\hat{s}^i$ de parties $\hat{Q}_i$ ne sont gardées que les parties satisfaisant :

$$\hat{Q}'_i = \left\{ \hat{q} \in \hat{Q}_i,\ \|\hat{q} - \hat{s}^*\| < t_d \right\}, \tag{2.27}$$

où $\hat{s}^*$ est l'ancre projetée de la partie $\hat{q}$. Le seuil t_d est matérialisé par le cercle rouge centré sur l'ancre dans la figure 2.21a, seules les parties dont l'ancre associée (extrémité de la flèche bleue) étant à l'intérieur du cercle sont associées à l'ancre. Le graphe étoilé $\hat{g}^i$ correspondant à une ancre $\hat{s}^i$ de parties associées $\hat{Q}'_i$ et l'ensemble des graphes étoilés prédits sont définis comme :

$$\begin{aligned} \hat{g}^i &= \hat{s}_i \cup \hat{Q}'_i \\ \hat{G} &= \{\hat{g}_i\}_{i \in 1 \ldots \hat{N}_n}, \end{aligned} \tag{2.28}$$

où $\hat{N}_n$ est le nombre de graphes étoilés prédit. La figure 2.21b illustre le graphe étoilé final reconstruit pour l'exemple de structure de plante donné.

Synthèse Pour résumer, dans un premier temps, les cartes de chaleur sont filtrées pour en extraire les maxima locaux ayant un score de confiance supérieur à un seuil t_c, donnant un ensemble de points d'intérêt (ancres et parties) décimés (de faible définition). Ensuite, les vecteurs de compensation de décimation sont ajoutés aux points d'intérêt afin de réduire l'erreur d'arrondi causée par la décimation. Finalement, les vecteurs d'indice de regroupement sont ajoutés aux parties et donnent un ensemble d'ancres projetées qui sont associées à l'ancre prédite la plus proche à condition que cette distance soit inférieure à un seuil t_d. Les graphes étoilés sont définis par l'ancre et les parties associées par ce processus.

La section 3.4 présente le plan d'entraînement de SDNet, des recherches exhaustives sur les paramètres t_c et t_d et une comparaison des performances avec YOLO sur les tâches de localisation de points d'intérêt et de comptage d'organes (nombre de feuilles). Le réseau SDNet, son paramétrage et son évaluation ont fait l'objet d'un article de conférence [77].

2.4 Suivi de plantes dans des séquences d'images basé sur les points d'intérêt de tige

Cette section présente un algorithme permettant de suivre les plantes détectées dans des séquences d'images en tirant parti de la redondance temporelle introduite par le recouvrement entre images successives. Dans l'exemple des séquences d'images captées par le module vision de BIPBIP, ce suivi a pour finalité d'augmenter la robustesse de la détection. En effet, le recouvrement spatial important entre plusieurs images consécutives permet d'observer plusieurs fois les mêmes plantes sous des angles de vue différents.

L'agrégation des plantes ne peut être réalisée que si l'on est capable de calculer le déplacement des plantes d'une image à l'autre pour pouvoir ensuite comparer les positions des plantes d'une image avec celles de l'autre image, ce qui n'est a priori pas un problème simple. Le profil des plantes peut fortement varier à cause de la perspective, phénomène d'autant plus marqué que la distance de la caméra à la scène est faible devant la taille et l'altitude des objets observés. Par ailleurs, le vent peut modifier leur forme, les plantes peuvent se recouvrir, être masquées ou cachées par des adventices, etc. Dans ces conditions il n'est pas évident d'estimer précisément le mouvement de chaque plante. Cependant, le bloc-outil BIPBIP et le module de vision opèrent dans des conditions bien particulières et les mouvements de la caméra sont contraints. Ces contraintes peuvent être exploitées pour réduire la complexité du problème et proposer une méthode robuste de suivi de plantes dans les séquences d'images.

L'algorithme de vision par ordinateur présenté dans cette section propose une méthode d'agrégation de plantes basée sur le suivi des points d'intérêt de bases de tige (point d'ancrage dans le sol) via l'estimation du mouvement de la caméra du module de vision de BIPBIP. La section 2.4.1 présente les hypothèses réalisées afin de pouvoir calculer de manière robuste le déplacement de la caméra puis la section 2.4.2 détaille un algorithme

itératif de calcul du déplacement et d'agrégation des détections de plante.

2.4.1 Hypothèses sur le mouvement et la pose de la caméra

Calculer le déplacement d'une caméra dans un environnement inconnu est un problème complexe, surtout lorsqu'il est basé sur un unique capteur (caméra couleur). De nombreuses perturbations de l'environnement rendent ce calcul non trivial : présence d'objets en mouvement ou d'objets peu texturés (matériaux fortement réflecteurs, noirs ou comportant des aplats unis de couleur), masquages et perturbations optiques (objets partiellement masqués, objets transparents, flou de mouvement) ou encore une forte parallaxe[13] ou perspective. Par ailleurs, peu de systèmes robotiques (agricoles notamment) utilisent une unique caméra pour estimer le déplacement dans l'espace. En général d'autres capteurs sont utilisés conjointement : LiDAR, caméra stéréoscopique, centrale inertielle, GPS, etc. Dans le domaine agricole, le GPS de haute précision (*Real Time Kinematic* (RTK) : cinématique en temps réel) est couramment employé car il propose une bonne précision en localisation. Une alternative est de simplifier le problème en réalisant des hypothèses sur le modèle de la scène observée puis en utilisant des algorithmes de vision par ordinateur pour estimer le déplacement.

Le RTK est, par exemple, employé pour guider un robot de désherbage de précision [48, 47]. Le robot BoniRob [25] utilise le RTK en plus d'un GPS classique pour le guidage dans le champ. Concernant les approches sans GPS, il est possible d'utiliser un système avec une caméra et un sonar puis d'employer des algorithmes d'odométrie visuelle (*Visual Odometry* (VO)) et de localisation et cartographie simultanées (*Simultaneous Localization and Mapping* (SLAM)) pour calculer la position et le déplacement d'un bloc-outil de désherbage de précision [136]. Cette approche nécessite néanmoins de faire l'hypothèse que la scène est statique (les objets ne bougent pas, pas de vent par exemple) et que la scène est suffisamment texturée.

Le bloc-outil BIPBIP ne possède qu'une caméra couleur pour la détection. Pour simplifier la conception, il n'y a pas de centrale inertielle et le système RTK éventuel du porteur n'est pas employé pour éviter d'en dépendre. Il est donc nécessaire d'utiliser une méthode d'estimation du mouvement basée sur la structure de l'image uniquement. Cependant, le bloc-outil BIPBIP opère dans des conditions bien particulières (voir section 1.1) qui permettent de simplifier la formulation du problème par le biais de quelques hypothèses :

- Le sol est majoritairement plan et les rangs de culture droits. La scène est globalement statique (pas de mouvement d'objets, le sol est statique mais les plantes peuvent bouger sous l'effet du vent).

[13] La parallaxe est l'impact du changement de la position de l'observateur (la caméra) sur ce qu'il perçoit (position et aspect des objets).

- La caméra est toujours face au sol et son axe optique étant perpendiculaire à la surface du sol[14].
- L'orientation de la caméra est fixe, l'axe horizontal de l'image étant parallèle à l'axe des rangs de cultures. Il n'y a donc pas de rotation de caméra possible.
- La hauteur de la caméra (distance du centre optique au sol) est constante et connue (35 cm sur le dernier prototype).
- Le bloc-outil avance suffisamment lentement pour que les images successives se superposent largement.

Ces conditions permettent de simplifier l'estimation du mouvement de la caméra dans l'espace. Elles imposent que le facteur d'échelle soit connu (i.e. l'aire réelle au sol couverte par un pixel) et que chaque pixel représente la même surface au sol (voir annexe C). Le corollaire est que la vitesse d'avancement de la caméra est proportionnelle à la vitesse apparente du mouvement du sol, et que celui-ci est identique en tout point du sol. En estimant ce déplacement, il est donc possible de projeter les points du sol de chaque image dans un référentiel commun, i.e. on peut connaître la position d'un point du sol d'une image I_n dans le référentiel d'une autre image I_m. En remarquant que la base de la tige des plantes est située au niveau du sol, il est possible d'estimer le déplacement des tiges de plantes dans les images successives, et donc des plantes entières dans le cas du détecteur de structure de plantes SDNet (voir section 2.3) puisqu'il détecte les plantes et leurs caractéristiques via leur tige. Néanmoins, suivre uniquement les bases de tiges et non la plante entière (dans le cas de YOLO par exemple, voir section 2.2) reste pertinent et permet d'augmenter la robustesse de la détection de ce point d'intérêt en vue de réaliser du désherbage de précision, ce qui est le cas d'application de BIPBIP. Dans la suite, on s'intéressera donc plus particulièrement au suivi de la base de tige plutôt qu'à celui de la plante entière.

En pratique, pour que l'estimation soit robuste, il faut prendre en compte plusieurs phénomènes parasites :

- Le sol n'est pas parfaitement plan. Il est donc préférable de prendre plusieurs mesures de l'estimation du déplacement à différents endroits de l'image.
- Les plantes ont une hauteur non nulle. Il faut donc les éliminer du calcul les surfaces où il y a des plantes afin de ne pas perturber l'estimation.

La section suivante présente un algorithme itératif qui estime ce mouvement de manière robuste et agrège les points d'intérêt de tige. Cet algorithme, son paramétrage et son évaluation ont fait l'objet d'une publication dans un article de revue [76].

[14] C'est le cas sur le dernier prototype. Les prototypes précédents requéraient un ajustement manuel.

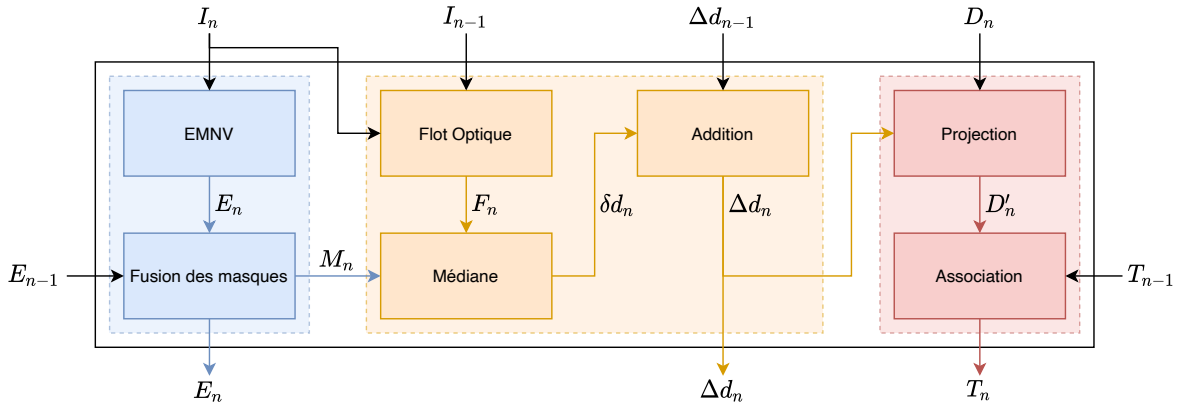


FIGURE 2.22 : Schéma du processus d'agrégation temporel itératif, composé de trois modules successifs : l'extraction du masque M_n du sol (en bleu) via un indicateur basé sur l'EGI [135] E_n, l'estimation du déplacement δd_n entre l'image I_n et I_{n-1} et du déplacement total Δd_n depuis la première image I_1 via le flot optique dense [37] F_n (en orange) et la projection des détections de points d'intérêt D_n puis, leur agrégation dans les trackers T_n via une variante de la métrique d'association de COCO [86] (en rouge).

2.4.2 Algorithme d'agrégation de points d'intérêt itératif

Le but de l'algorithme d'agrégation est de rendre plus robuste l'estimation de la position des bases de tiges de plantes. Son principe est, par l'estimation du déplacement de la caméra, de projeter les détections de base de tige dans un référentiel spatial commun afin d'associer les détections qui représentent le même objet dans différentes images. Pour garantir un fonctionnement en temps réel, un algorithme itératif causal, illustré dans la figure 2.22, est présenté dans cette section. Celui-ci opère donc image par image. À chaque instant n, il prend en entrée deux données : l'image I_n et des détections de points d'intérêt de base de tige D_n prédits dans l'image I_n par l'un des détecteurs présentés dans les sections 2.2 et 2.3. Il prédit un ensemble de *trackers* T_n qui représentent les détections *agrégées*, i.e. projetées dans un référentiel spatial commun et groupées par détections décrivant la même plante. Un *tracker* représente donc une unique tige de plante et les multiples observations de celle-ci dans les différentes images de la séquence. Les autres variables décrites dans la figure 2.22 (E_{n-1}, I_{n-1}, Δd_{n-1} et T_{n-1}) sont des variables transitoires qui sont mises à jour à chaque itération puis transférées pour la prochaine itération.

L'algorithme est construit autour de trois modules séquentiels :

- **L'extraction du masque du sol :** les hypothèses réalisées sur le mouvement de la caméra impliquent que seuls les points du sol satisfont les conditions, ce module permet donc d'extraire un masque M_n des pixels du sol de l'image I_n (illustré en bleu sur la figure 2.22).

- **L'estimation du déplacement :** le déplacement de la caméra δd_n entre l'image I_n et l'image précédente I_{n-1} puis le déplacement total Δd_n depuis la première image

I_1 sont calculés de manière robuste via le calcul du flot optique dense F_n des pixels du sol (illustré en orange sur la figure 2.22).

- **L'agrégation des détections :** les points d'intérêt sont projetés dans un référentiel spatial commun puis un algorithme d'association lie les détections de plusieurs images représentant la même plante (illustré en rouge sur la figure 2.22) : ce sont les trackers T_n.

2.4.2.1 Extraction du masque du sol

L'estimation du mouvement de la caméra est réalisé à partir des points dont la profondeur (distance à l'objectif) est connue. Or ceci n'est valable que pour les points du sol d'après les hypothèses formées précédemment. Le but de cette étape est donc de calculer un masque permettant d'identifier les pixels du sol dans l'image I_n. Ce masque $M_n \in \{0,1\}^{W \times H}$ vaut 1 pour chaque pixel du sol et 0 pour les autres pixels. L'algorithme opère en deux étapes illustrées en bleu sur la figure 2.22 :

- **L'Extraction du Masque des Non Végétaux (EMNV) :** cette première étape extrait le masque des pixels de non-végétaux E_n de l'image I_n par le biais d'un indicateur colorimétrique calculé sur l'image.

- **La fusion des masques :** cette seconde étape fusionne le masque E_n avec le masque E_{n-1} de la précédente image I_{n-1} puis applique des opérations de traitement d'images afin de rendre l'estimation des pixels du sol plus robuste.

La première étape de cet algorithme est l'extraction du masque E_n des pixels non-végétaux de l'image I_n, c'est-à-dire de tous les pixels qui représentent le sol, les outils de travail du sol ou la bâche de protection du module de vision de BIPBIP. Une manière de procéder couramment employée est l'utilisation de l'indice colorimétrique Excess Green Index (EGI) [135] permettant d'accentuer la composante verte d'une image en couleur. L'EGI G_n d'une image I_n est défini pour tout pixel de coordonnées (x, y) de l'image I_n par :

$$G_n(x,y) = 2 \times I_n^G(x,y) - I_n^R(x,y) - I_n^B(x,y), \tag{2.29}$$

où I_n^R, I_n^G et I_n^B sont respectivement les canaux rouge, vert et bleu de l'image I_n. Ceci a pour effet de mettre en avant les pixels de végétation qui ont donc une valeur d'intensité élevée dans l'EGI. Il est ensuite simple de segmenter les pixels qui n'appartiennent pas à la végétation en binarisant l'EGI par le biais d'un seuil t_e : les pixels de l'EGI qui ont une valeur supérieure au seuil t_e sont considérés comme des pixels de végétation. Le masque des non-végétaux E_n est donc défini en tout point de coordonnées (x, y) par :

$$E_n(x,y) = \begin{cases} 1 & \text{si } G_n(x,y) < t_e \\ 0 & \text{sinon,} \end{cases} \tag{2.30}$$

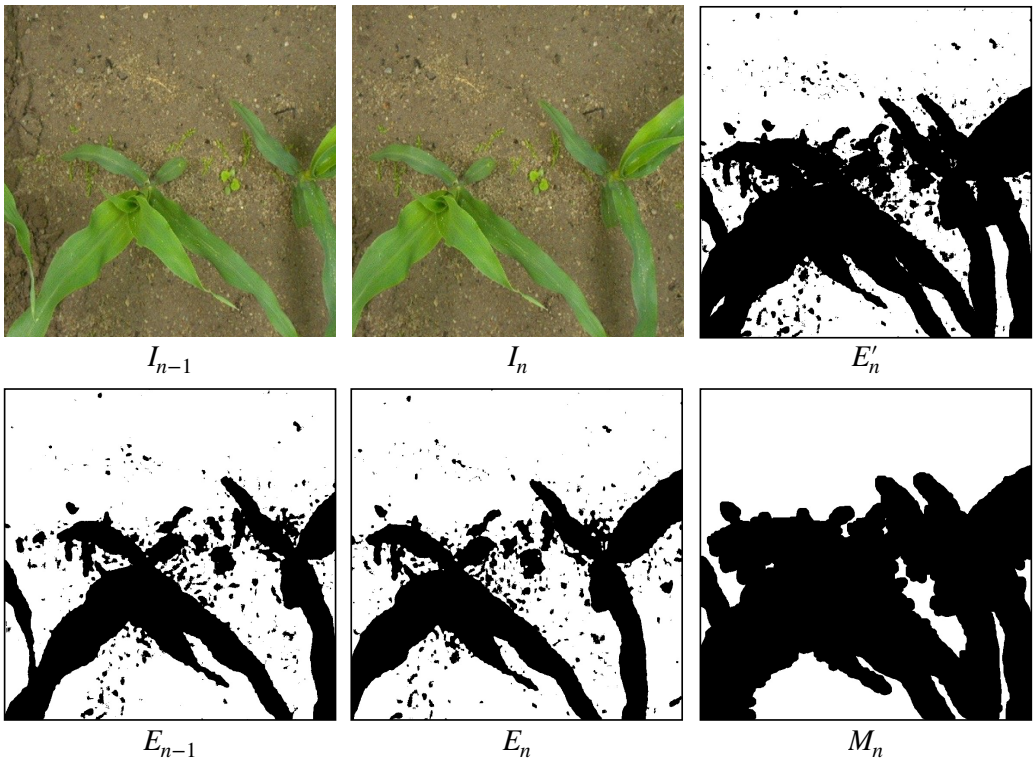


FIGURE 2.23 : Illustration de l'étape d'extraction du masque du sol sur deux images successives I_{n-1} et I_n avec les masques des non-végétaux E_{n-1} et E_n, le masque fusionné E'_n et le masque filtré final des pixels du sol M_n. La cadence de prise de vue est volontairement abaissée pour améliorer la visualisation. Les pixels en blanc représentent les pixels des non-végétaux et les pixels en noir les pixels des végétaux.

où $t_e \in \mathbb{R}$ est le seuil sur l'intensité de l'EGI. La figure 2.23 illustre deux masques E_{n-1} et E_n pour deux images successives. Les pixels de la végétation sont représentés en noir et les pixels non-végétaux sont représentés en blanc. La valeur de t_e est choisie qualitativement (voir partie 3.3.3). Une valeur trop haute pouvant inclure des pixels de la végétation dans le masque du sol alors qu'une valeur trop basse risque d'écarter trop de pixels du sol, considérés à tort comme de la végétation.

La seconde étape de l'algorithme consiste en une fusion du masque E_n avec le masque des non-végétaux E_{n-1} de l'image précédente et en une opération morphologique de filtrage. En effet, la forte parallaxe causée par la taille importante des plantes entraîne des zones d'occultation et de masquage à la bordure des feuilles et peut provoquer un mouvement apparent rapide des feuilles (plus un objet est proche de la caméra, plus son mouvement apparent est grand), ce qui peut se révéler problématique pour l'algorithme de calcul du flot optique présenté dans la section suivante, occasionnant des valeurs aberrantes. Afin d'éliminer ces zones, le masque des non-végétaux E_n est fusionné avec le masque E_{n-1} de l'image précédente, i.e. pour tout pixel de coordonnées (x, y) :

$$E'_n(x, y) = E_{n-1}(x, y) \times E_n(x, y). \tag{2.31}$$

Autrement dit, le masque fusionné E'_n comporte un 1 aux coordonnées où il n'y a pas de végétation dans les deux images successives I_n et I_{n-1} et un 0 aux autres coordonnées (illustré dans la figure 2.23 en haut à droite). Ceci à pour effet de supprimer les zones qui sont considérées comme "invalides" (présence de végétation) dans au moins l'une des deux images. Une dernière étape de filtrage est l'application d'un opérateur morphologique `morph` (ouverture morphologique, remplissage de trous, suppression des petits objets) qui agrandit légèrement la surface des zones du masque E'_n où il n'y a pas de végétation afin de réserver une marge supplémentaire aux bords des feuilles, zones qui peuvent être fortement perturbées par le vent par exemple. Pour finir, un masque H de la zone de sûreté des outils[15] est additionnellement appliqué. Le masque du sol final est donc défini comme :

$$M_n = \text{morph}(E'_n) \times H. \tag{2.32}$$

La figure 2.23 illustre le masque du sol final (sans le masque H, non représenté pour simplifier l'illustration) en bas à droite.

2.4.2.2 Calcul du déplacement

Sous les hypothèses énoncées, le déplacement de la caméra entre les instants $n-1$ et n est directement proportionnel à la vitesse de déplacement des points du sol, et ceux-ci ont une valeur égale en tout point du sol. Toujours selon les hypothèses réalisées, le déplacement de la caméra se fait exclusivement dans un plan parallèle au plan du sol. Le vecteur de déplacement de la caméra entre deux instants n et $n-1$ est donc un vecteur à deux dimensions $\delta d_n = (\delta d_n^x, \delta d_n^y) \in \mathbb{R}^2$ exprimé dans le référentiel $\mathcal{R}_n = (X_n, Y_n)$ de l'image I_n et qui a pour origine le coin haut-gauche de l'image (voir annotations du référentiel dans la figure 2.24 pour la première image I_1).

Estimer la vitesse de déplacement d'un point entre deux images peut être réalisé via un algorithme de calcul du flot optique. Celui-ci désigne un ensemble de méthodes de vision par ordinateur, qui pour deux images présentant la même scène sous deux angles différents mais proches, sont capables d'estimer le mouvement de différents points de l'espace entre les deux instants. Ils sont, par exemple, utilisés pour estimer le déplacement d'un objet entre deux images. Dans le cas qui nous intéresse, le flot optique peut être utilisé pour calculer le déplacement de points du sol.

Afin de générer une valeur de déplacement du sol robuste, il est pertinent d'estimer le déplacement d'un grand nombre de points du sol puis de moyenner ces valeurs puisqu'elles sont idéalement identiques en tous points. Une valeur de déplacement du sol peut être obtenue pour chaque pixel d'une image par la méthode de calcul du flot optique dense de Farnebäck [37]. Pour chaque pixel de coordonnée (x, y) dans l'image I_n, cette méthode

[15] H contient un 1 dans les zones où il n'y a pas d'outils (socs) ou de morceaux de la bâche de protection du module de vision visible et un 0 dans le cas contraire.

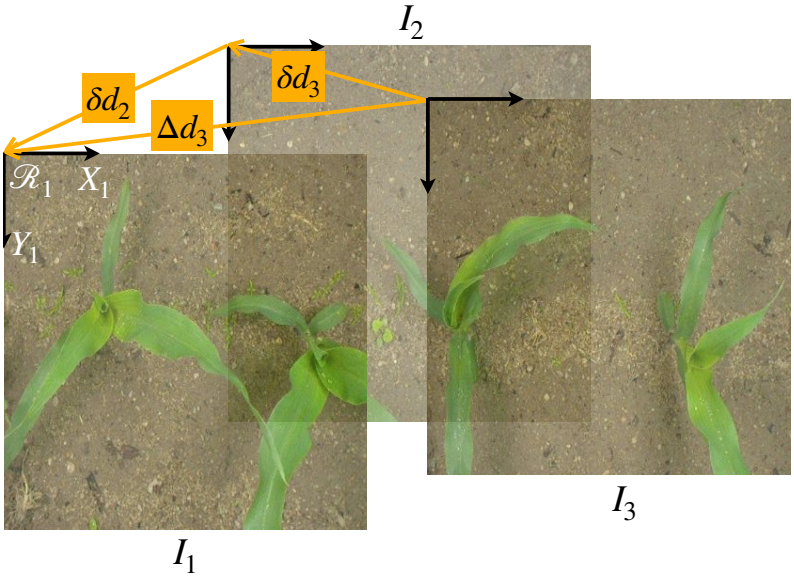


FIGURE 2.24 : Illustration de l'étape du calcul du déplacement du sol présenté sur trois images successives (la cadence de prise de vue est volontairement abaissée pour améliorer la visualisation). Les images sont superposées en transparence et translatées de la valeur du déplacement du sol (les bases de tiges de la même plante sont donc confondues). Les vecteurs de déplacement du sol entre images δd_2 et δd_3 et le vecteur de déplacement total Δd_3 depuis la première image sont matérialisés par des flèches oranges.

calcule sa position probable dans l'image I_{n-1} en comparant un bloc de pixels autour de (x, y) avec d'autres blocs voisins de cette position dans l'image I_{n-1}.

Cette méthode fonctionne mieux si les blocs sont très texturés et que le mouvement entre les deux images est de petite amplitude. D'une part, un bloc très texturé permet de réduire la probabilité d'identifier plusieurs blocs similaires (un aplat de couleur comme le ciel par exemple). D'autre part, les algorithmes de calcul du flot optique ne réalisent pas de recherches exhaustives dans toute l'image mais seulement sur un voisinage afin de limiter le temps de calcul. Un mouvement de petite amplitude permet de limiter la recherche des blocs dans le voisinage. Dans le cas du bloc-outil BIPBIP, ces deux conditions sont remplies : (i) le module avance assez lentement et la cadence d'acquisition d'images est assez élevée pour que le recouvrement entre deux images successives soit suffisant, ce qui rend le déplacement du sol plus faible, et (ii) le sol est généralement très texturé (cailloux, mottes de terres, petites herbes, etc.).

Le calcul du déplacement est réalisé en trois étapes illustrées en orange sur le schéma de la figure 2.22. D'abord, le flot optique $F_n \in \mathbb{R}^{W \times H \times 2}$, un champ vectoriel à deux dimensions de même taille que l'image I_n et exprimé dans le référentiel $\mathcal{R}_n$, est estimé via la fonction `flot_optique`[16] qui représente le calcul du flot optique dense de Farnebäck :

$$F_n = \text{flot\_optique}(I_n, I_{n-1}). \tag{2.33}$$

[16]L'implémentation de la librairie OpenCV est sélectionnée.

Ce champ vectoriel indique le déplacement subi par tous les pixels de l'image, végétation comprise, entre les deux instants. Il faut ensuite le filtrer avec le masque du sol M_n précédemment établi, seuls les vecteurs présents aux coordonnées où M_n est égal à 1 sont conservés. La liste des vecteurs de déplacement des pixels du sol $F'_n \in \mathbb{R}^{N_v \times 2}$ s'exprime comme :

$$F'_n = \{F_n(x,y) \,|\, M_n(x,y) = 1\}_{x \in 1 \ldots W,\ y \in 1 \ldots H}\,, \tag{2.34}$$

où N_v est le nombre de vecteurs conservés, i.e. le nombre de pixels de valeur 1 dans M_n. Le déplacement de la caméra δd_n (en pixels) entre les deux images I_{n-1} et I_n est ensuite obtenu en prenant la médiane, notée `med`, des vecteurs sur chaque coordonnée :

$$\delta d_n = (\mathrm{med}(F'_{x,n}),\ \mathrm{med}(F'_{y,n})), \tag{2.35}$$

où $F'_{x,n}$ est la liste des composantes sur l'axe horizontal des vecteurs F'_n et $F'_{y,n}$ celle sur l'axe vertical. La médiane est choisie plutôt que la moyenne car elle est plus robuste aux éventuelles valeurs de flot optique aberrantes. La figure 2.24 illustre par des flèches jaunes les vecteurs de déplacement du sol δd_2 et δd_3 entre trois images successives (la cadence de prise de vues a été artificiellement diminuée pour plus de clarté). Les images sont superposées en transparence et translatées de la valeur du déplacement du sol entre images afin d'illustrer l'alignement des bases tiges de plante que cette méthode produit. Finalement, le déplacement total Δd_n du sol depuis la première image I_1 est obtenue par intégration des déplacements successifs :

$$\Delta d_n = \sum_{i=1}^{n} \delta d_n, \tag{2.36}$$

avec la condition initiale $\Delta d_1 = 0$. Sur la figure 2.24, le vecteur de déplacement du sol total Δd_3 est illustré par une flèche jaune allant de l'image I_3 à l'image I_1. En pratique, la valeur du déplacement réel de la caméra en mètres n'est pas nécessaire pour l'agrégation des points d'intérêt et le déplacement du sol en pixels suffit. Néanmoins, il peut être calculé en multipliant le déplacement du sol par le facteur d'échelle $f_v = \frac{W}{W_r}$ où W est le nombre de pixels sur l'axe horizontal et W_r la distance visible du rang sur l'image.

2.4.2.3 Agrégation des détections

L'agrégation des détections, illustrée en rouge sur la figure 2.22, est un processus réalisé en deux étapes :

- **L'étape de projection :** Les détections D_n, exprimées dans le référentiel de l'image I_n, sont projetées dans le référentiel commun défini comme celui de la première image $\mathcal{R}_1$, ce qui permet par la suite d'appliquer un algorithme de regroupement des détections selon un critère de proximité spatiale.

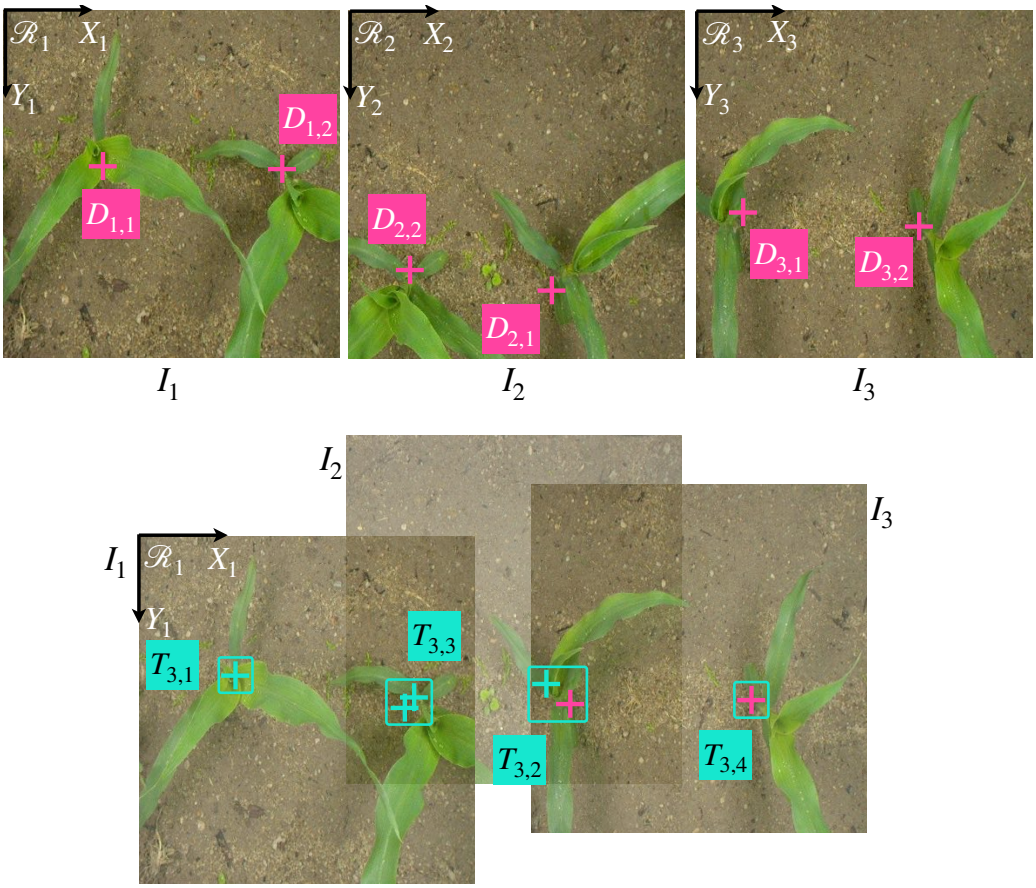


FIGURE 2.25 : Illustration de l'étape d'association des détections présentée sur trois images consécutives (la cadence de prise de vue est volontairement abaissée pour améliorer la visualisation). Les détections de points d'intérêt D_1, D_2 et D_3 (bases de tige de plante) sont présentées sur la ligne du haut (croix roses) et l'ensemble des trackers T_3 calculés à l'instant $n = 3$ est illustré sur la ligne du bas (boîtes bleues). Les détections en rose dans les trackers correspondent aux détections qui viennent d'être associées : $D_{3,1}$ a été associée au tracker $T_{3,2}$ et $D_{3,4}$ et créée un nouveau tracker $T_{3,4}$ en l'absence de tracker associé. Les autres trackers n'ont pas été mis à jour en l'absence de détections associées.

- **L'association des détections :** Un algorithme d'agrégation glouton [28] (Greedy Algorithm) associe chaque détection projetée aux détections passées qui représentent la même plante, i.e. au tracker de la plante.

Pour rappel, les détections D_n consistent en un ensemble de prédictions de points d'intérêt de base de tige de plante. Chaque point d'intérêt $D_{n,i} \in D_n$ est caractérisé par sa position (x_{ni}, y_{ni}) dans l'image I_n, son score de confiance $l_{n,i}$ et sa classe (tige de maïs, tige de haricot) c_{ni} :

$$D_n = \{D_{n,i} = (x_{ni}, y_{ni}, c_{ni}, l_{ni})\}_{i \in 1 \ldots N_n} , \tag{2.37}$$

où N_n est le nombre de détections de plante dans l'image I_n. La figure 2.25 illustre trois images successives avec les détections de tige de maïs en rose (la cadence de prise de vue est volontairement abaissée pour améliorer la visualisation). Les détections D_n étant exprimées dans le référentiel $\mathcal{R}_n$, la première étape est de les projeter dans le référentiel

Algorithme 1 Algorithme d'association des détections projetées D'_n aux trackers T_{n-1}.

Inputs : T_{n-1}, D'_n
Parameters : $max_{inactive}$, max_{dist}
Outputs : T_n
$trackers \leftarrow$ `filterInactive`$(T_{n-1}, max_{inactive})$
$detections \leftarrow$ `sortByDecreasingConfidence`(D'_n)
$matches \leftarrow$ **new** `list`
for $detection \in detections$ **do**
  $(bestDistance, bestTracker) \leftarrow (+\infty, nil)$
  for $tracker \in trackers$ **do**
    $distance \leftarrow \|$`barycenter`$(tracker) - detection\|_2$
    if $distance < bestDistance$ **then**
      $(bestDistance, bestTracker) \leftarrow (distance, tracker)$
    end if
  end for
  if $bestDistance < max_{dist}$ **and** $bestTracker \notin matches$ **then**
    $matches \leftarrow matches \cup (detection, bestTracker)$
  end if
end for
$T_n \leftarrow matches \cup$ `notAssociated`$(T_{n-1}) \cup$ `notAssociated`$(detections)$
return T_n

$\mathcal{R}_1$ en ajoutant le déplacement du sol total Δd_n depuis la première image aux coordonnées des points d'intérêt afin d'obtenir les détections translatées D'_n :

$$D'_n = \{(x_{ni} + \Delta d_n^x, y_{ni} + \Delta d_n^y, c_{ni}, l_{ni})\}_{i \in 1 \ldots N_n}, \tag{2.38}$$

où Δd_n^x est la composante sur l'axe horizontal X_n du vecteur de déplacement total et Δd_n^y celui sur l'axe vertical Y_n. La figure 2.25 illustre sur la ligne du bas les détections D_3 de l'image I_3 projetée dans le référentiel $\mathcal{R}_1$ (croix roses).

La seconde partie de l'algorithme d'agrégation temporelle est l'association des détections aux trackers T_{n-1} de l'itération précédente représentant les plantes déjà détectées par le passé. Le but est ici de trouver l'association optimale entre les détections projetées D'_n et les trackers T_{n-1}. L'algorithme proposé à cette fin est un algorithme dit *glouton* [28] : il associe itérativement chaque détection à un tracker selon une heuristique locale qui peut ne pas correspondre à l'optimum global. Ces algorithmes sont généralement plus rapides que les recherches exhaustives puisqu'ils ne traitent pas toutes les configurations possibles. L'algorithme d'association proposé est basé sur celui employé dans la métrique d'évaluation de la détection d'objets COCO [86] qui est chargé de faire correspondre les boîtes englobantes prédites avec les vérités de terrain.

La procédure de cet algorithme est formalisée par le pseudo-code de l'algorithme 1. Il fonctionne de la sorte :

- Les trackers T_{n-1} sont d'abord filtrés pour ne garder que ceux qui sont *actifs*. Un

tracker est dit actif si une détection lui a été associée lors des $max_{inactive}$ itérations précédentes de l'algorithme d'association. Cette étape permet de réduire le nombre de trackers candidats à considérer lors de l'association et, donc, de limiter le temps de calcul. Cela supprime, par exemple, les trackers qui ne sont plus visibles dans l'image.

- Les détections D'_n sont ensuite triées par ordre décroissant de score de confiance. Cela permet à la procédure d'association de traiter en priorité les détections jugées les plus fiables.

- Une liste d'appariements `matches` est initialisée à un état vide. Celle-ci est remplie progressivement avec les couples détection-tracker associés pendant la procédure d'association. Cette liste est renvoyée et correspond aux nouveaux trackers T_n.

- Le cœur de la procédure d'association est constitué de deux boucles imbriquées, l'une sur les détections et l'autre sur les trackers. L'association est basée sur un critère de proximité spatiale entre une détection et un tracker. Un tracker représentant un groupe de détections de points d'intérêt, sa position est estimée par la moyenne des positions des détections qu'il contient. La distance d'une détection à un tracker est donc la distance de la détection au barycentre du tracker. Lors de cette procédure, pour chaque détection, le tracker le plus proche de celle-ci est calculé. Ensuite, la détection est associée au tracker si les conditions suivantes sont remplies : (i) le tracker n'est pas déjà associé à une autre détection (i.e. il n'est pas déjà présent dans la liste `matches`) et (ii) si la distance au tracker est inférieure à une distance minimale max_{dist}.

- Finalement, les nouveaux trackers T_n sont définis comme l'union : (i) des trackers mis à jour ($matches$), (ii) des trackers non associés `notAssociated`($trackers$), et (iii) des trackers initialisés par les détections non associées `notAssociated`(T_{n-1}).

La figure 2.25 illustre la procédure d'agrégation sur trois images successives comportant deux annotations D'_3 (croix roses sur la ligne du bas). La première détection $D'_{3,1}$ (à gauche) est associée au tracker déjà existant $T_{3,2}$ (qui contenait une seule détection, $D'_{2,1}$) car il s'agit du plus proche. La seconde détection $D'_{3,2}$ est, elle aussi, la plus proche du tracker $T_{3,2}$. Cependant, celui-ci vient déjà d'être associé, un nouveau tracker $T_{3,4}$ est créé pour cette détection. Les autres trackers ne sont pas mis à jour (leur durée d'inactivité augmente donc de 1).

Les trackers T_n établis par cette procédure peuvent être filtrés sur la base du nombre d'observations. Par exemple, un tracker avec un nombre élevé d'observations est probablement plus fiable qu'un tracker avec seulement une ou deux observations. Une méthode de filtrage permettant de tirer parti de cette redondance d'information pour augmenter la robustesse de la détection est de ne considérer comme valides que les trackers ayant un

nombre d'observations supérieur à un seuil min_{dets}, les autres étant considérés comme des faux positifs :

$$T_{n,valid} = \{T_{n,i} \in T_n, \texttt{nbDets}(T_{n,i}) > min_{dets}\}\,, \tag{2.39}$$

où `nbDets` est la fonction retournant le nombre de détections d'un tracker. Le gain de performance apporté par cette stratégie de filtrage ainsi que l'influence des valeurs de $max_{inactive}$, max_{dist} et min_{dets} sur la performance du filtrage sont analysés dans le chapitre 3.4.

2.5 Synthèse des contributions

Ce chapitre a développé, d'une part, deux méthodes d'apprentissage profond permettant de détecter des plantes d'intérêts dans des images et de collecter des propriétés sur celles-ci, comme la position des organes et le nombre de feuilles et, d'autre part, une méthode de suivi de plantes dans des séquences d'images afin d'améliorer la robustesse de la détection.

Une première partie a d'abord développé les hypothèses et la modélisation du problème. Notamment, un modèle simplifié de la plante est proposé : celle-ci est identifiée par une boîte englobante permettant de la localiser dans une image, par une classe indiquant la variété de la plante (maïs, haricot, etc.) et par des points d'intérêt localisant les organes principaux de la plante, comme la tige et les feuilles. Ce modèle a ensuite été étendu au cas des séquences temporelles d'images. Des hypothèses sur le déplacement de la caméra du module vision de BIPBIP, cadre dans lequel les algorithmes sont exploités pour réaliser du désherbage de précision, ont été faites afin de formaliser la méthode de suivi des plantes envisagée.

La deuxième partie a proposé une méthode innovante basée sur un réseau de neurones de détection d'objets afin de localiser les organes de plantes et de compter le nombre de feuilles. Elle a détaillé les différentes architectures de détecteurs d'objets et justifié le choix d'une famille de réseaux adaptée aux contraintes d'exécution en temps réel du bloc-outil de désherbage mécanique BIPBIP. Elle a finalement présenté l'architecture adoptée qui concerne les réseaux de la famille YOLO.

La troisième partie a défini la structure de plante, un modèle basé sur un graphe en étoile décrivant la position des organes d'une plante et leurs relations, puis proposé SDNet, une architecture neuronale inspirée des détecteurs de pose capable de localiser les plantes et de détecter un nombre arbitraire de feuilles par plante. La conception et les considérations architecturales, notamment sur le respect de la contrainte temps réel, ont ensuite été détaillées.

Une quatrième partie a présenté une méthode de vision par ordinateur pour suivre des plantes dans des séquences d'images afin d'augmenter la robustesse de la détection grâce à la redondance temporelle présente dans les images. D'abord, des hypothèses sur les caractéristiques des images et des mouvements de la caméra ont été réalisées puis

une méthode de suivi de points d'intérêt basée sur l'estimation du déplacement de la caméra via le flot optique et sur un algorithme itératif d'agrégation de points d'intérêt a été détaillée.

Le chapitre suivant introduit les protocoles expérimentaux mis en place afin d'évaluer la pertinence des choix algorithmiques et de comparer les approches entre elles.

Chapitre 3

Protocoles et résultats

Sommaire

L'objectif du travail de ce mémoire est de proposer des méthodes de vision par ordinateur et d'apprentissage profond pour des applications en agriculture de précision, et notamment en désherbage de précision. Dans ce but, le chapitre 2 a présenté des solutions pour détecter des plantes, collecter certaines propriétés telles que la position des organes et le nombre de feuilles, et améliorer la qualité des détections en tirant parti de la redondance temporelle des séquences d'images acquises par le module de désherbage de précision BIPBIP. Évaluer ces multiples aspects nécessite la définition de métriques d'évaluation tenant compte des différents facteurs de performance. La première section de ce chapitre introduit donc les tâches évaluées et les métriques d'évaluation choisies pour mener les expérimentations. Plus spécifiquement, deux tâches sont identifiées :

- La localisation d'organes de plantes.
- L'estimation du nombre de feuilles.

Des métriques pour évaluer les performances sur ces deux tâches sont ensuite détaillées.

Les réseaux de neurones profonds présentés dans les sections 2.2 et 2.3 qui nécessitent une grande quantité de données annotées pour l'apprentissage puisque ce sont des méthodes supervisées. Des données annotées sont aussi nécessaires pour la validation des choix de paramétrisation et l'évaluation des algorithmes. La deuxième section présente donc les différentes bases de données créées et les méthodes d'annotation établies puis détaille les méthodes d'entraînement des réseaux de neurones sur ces bases de données. Trois bases

de données sont introduites : une première pour l'entraînement et l'optimisation de la méthode YOLO (présentée dans la section 2.2), une deuxième pour l'entraînement de SDNet (présenté dans la section 2.3) et la comparaison des performances avec la méthode YOLO, et finalement, une troisième pour la validation des performances du suivi temporel des tiges de plante (présenté dans la section 2.4).

Toutes les méthodes présentées dans la section 2 dépendent de paramètres ou d'hyper-paramètres influant sur leurs performances et dont la valeur optimale n'est pas connue analytiquement. Néanmoins, des recherches empiriques peuvent être menées pour trouver un optimum local de ces paramètres, notamment au travers de recherches exhaustives comparant les performances pour une large gamme de valeurs. La troisième section présente donc des recherches empiriques sur l'impact de certains paramètres des algorithmes sur les performances et analyse les causes potentielles. Les recherches exhaustives sont par nature limitées et coûteuses puisque, d'une part, elles ne permettent que de trouver un optimum local et, d'autre part, elles nécessitent l'exécution d'un grand nombre d'expérimentations gourmandes en ressources de calcul, surtout lorsque ces expérimentations requièrent des entraînements répétés de réseaux de neurones profonds. Cette section se concentre uniquement sur les paramètres principaux, comme la définition des images d'entrée et le seuil de confiance des réseaux de neurones ou les paramètres de l'algorithme d'agrégation de points d'intérêt.

Finalement, une fois les paramètres optimaux établis pour les différents algorithmes, il est possible d'évaluer les performances. La quatrième section présente donc une comparaison des performances des deux réseaux de neurones sur les deux tâches évaluées puis valide l'amélioration des performances apportée par l'algorithme de suivi temporel de tige de plante. La dernière section synthétise et discute ensuite l'ensemble des résultats établis dans le cadre du travail proposé.

3.1 Tâches et métriques d'évaluation

Cette section détaille les tâches sélectionnées pour évaluer les algorithmes puis présente les métriques chargées de fournir les indicateurs de performance. Dans ce travail, deux méthodes de détection et de caractérisation de plantes ont été développées, ainsi qu'une méthode de suivi temporel de tiges :

- Une méthode basée sur YOLO, présentée dans la section 2.2, qui exploite de manière innovante un réseau de détection d'objets pour localiser des points d'intérêt et pour compter le nombre de feuilles des plantes.

- Une méthode basée sur SDNet, présentée dans la section 2.3, qui est une architecture dédiée à la détection de structure de plantes, i.e. la localisation des organes (tige et feuilles) et leurs relations.

- Une méthode d'agrégation de tiges de plantes, présentée dans la section 2.4, qui permet d'améliorer la détection des tiges dans des séquences d'images issues du module vision de BIPBIP.

Elles ont été développées afin de répondre à une problématique de désherbage de précision et pour explorer des applications potentielles en évaluation du stade de développement des cultures. Les tâches sélectionnées doivent donc permettre d'évaluer ces aspects. La première section 3.1.1 présente les deux tâches définies et la section 3.1.2 introduit les métriques permettant de les évaluer.

3.1.1 Tâches évaluées

Deux tâches sont définies, la première évalue une capacité relative au désherbage de précision et l'autre une capacité relative au développement des cultures.

3.1.1.1 Localisation d'organes de plantes

Cette première tâche évalue la capacité des algorithmes à détecter précisément la localisation des organes des plantes. Le désherbage de précision requiert une détection fine et précise des plantes pour pouvoir mener une action localisée sur les cultures. Dans le cas du bloc outil BIPBIP, c'est la position des tiges de plante qui est nécessaire pour pouvoir désherber mécaniquement l'intra-rang. Cette tâche propose donc d'évaluer la précision de la détection des points d'intérêt de tige de plante (maïs, haricot et poireau) mais aussi de feuilles.

Plusieurs facteurs rentrent en compte pour l'évaluation de cette tâche :

- La détection qui désigne l'aptitude à détecter le nombre correct de points d'intérêt dans une image, i.e. limiter le nombre de faux positifs et de faux négatifs.
- La localisation qui désigne l'aptitude à localiser convenablement les points d'intérêt.
- La classification qui désigne l'aptitude à prédire le type correct des points d'intérêt.

Il s'agit d'une tâche multifactorielle nécessitant une métrique d'évaluation prenant en compte les différents aspects influençant la performance. La figure 3.1 illustre cette tâche pour une image. La vérité de terrain est à gauche et la prédiction à droite. Il y a trois types de points d'intérêt : les tiges de maïs (en rouge), les tiges de haricot (en bleu) et les feuilles (en orange). La figure illustre cinq types d'erreurs : une erreur de localisation (1), deux erreurs de classification (2, 5), un oubli de détection (3) et un faux positif (4) sur une adventice. Les métriques présentées dans la section suivante permettent de quantifier précisément ces erreurs.

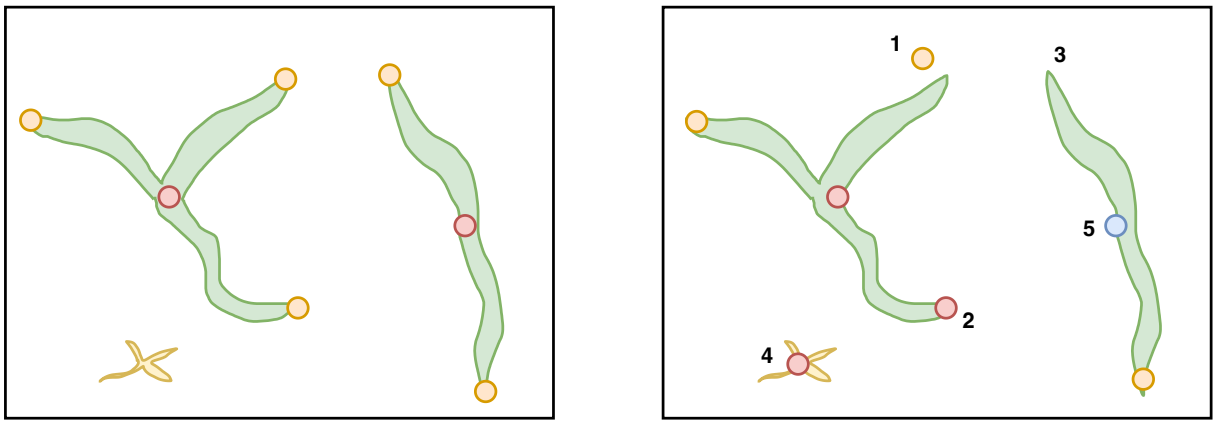


FIGURE 3.1 : Illustration de la tâche de localisation d'organes de plante sur une image. La vérité de terrain est à gauche et la prédiction à droite. Les points d'intérêt de tige de maïs sont en rouge, ceux de tige de haricot en bleu et les feuilles en orange. Cinq erreurs sont illustrées : une erreur de localisation (1), deux erreurs de classification (2, 5), un oubli de détection (3), et un faux positif (4) sur une adventice.

3.1.1.2 Estimation du nombre de feuilles

Cette seconde tâche évalue la capacité des algorithmes à détecter les plantes et à estimer le nombre d'organes qu'elles possèdent. Cette tâche peut permettre d'évaluer certains aspects de l'état des cultures, comme le stade de développement. Dans le cadre du travail proposé, la feuille est l'organe qui a été sélectionné car il s'agit d'un indicateur pertinent pour décrire le stade de développement. La tâche proposée consiste donc à estimer correctement le nombre de feuilles des plantes. Contrairement à d'autres travaux portant sur l'estimation du nombre de feuilles, comme [97, 45, 65], le travail proposé dans ce manuscrit ne se limite pas à une tâche de classification d'une seule plante. En effet, les méthodes développées permettent de détecter plusieurs plantes par image et donc de classifier plusieurs plantes en même temps. La tâche proposée évalue donc la capacité de classification et la capacité de détection.

Comme pour la première tâche, c'est une analyse multifactorielle prenant en compte :

- La détection qui désigne l'aptitude à détecter le nombre correct de plantes.
- La classification de la variété de plante qui évalue l'aptitude à prédire le type de plante (maïs, haricot, poireau).
- L'estimation du nombre de feuilles qui, comme son nom l'indique, désigne l'aptitude à prédire le nombre correct de feuilles.

La figure 3.2 illustre la tâche d'estimation du nombre de feuilles sur une image. La vérité de terrain est à gauche et la prédiction à droite. La lettre désigne le type de culture ("M" pour maïs, "H" pour haricot et "P" pour poireau) et le nombre désigne le nombre de feuilles. Une erreur d'estimation du nombre de feuilles (une en trop) est illustrée sur la plante de gauche, une erreur de classification de la variété de plante (haricot au lieu de

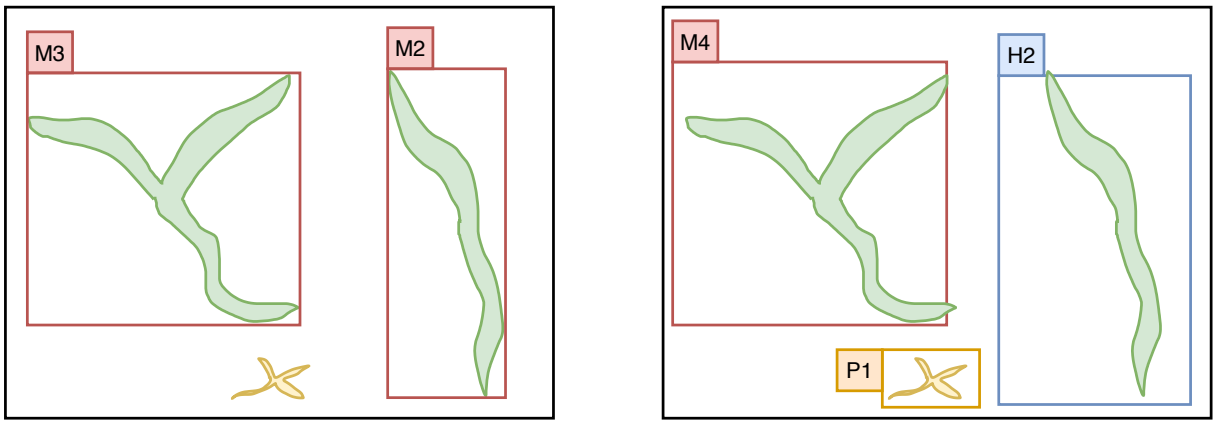


FIGURE 3.2 : Illustration de la tâche d'estimation du nombre de feuilles. La vérité de terrain est présentée à gauche et la prédiction à droite. La lettre désigne la variété de la plante ("M" pour maïs, "H" pour haricot et "P" pour poireau) et le nombre désigne le nombre de feuilles. Trois erreurs sont illustrées : une erreur d'estimation du nombre de feuilles (plante de gauche), une erreur de classification de la variété de la plante (plante de droite) et une erreur de détection (adventice en orange).

maïs) est présentée sur la plante de droite et une erreur de détection (faux positif) est identifiée par la boîte jaune sur l'adventice.

Il faut garder à l'esprit que la capacité de détection est évaluée pour chacune des deux tâches et que c'est le facteur ayant le plus d'impact sur la performance puisqu'il s'agit d'un prérequis avant de pouvoir évaluer les capacités de localisation et d'estimation du nombre de feuilles, i.e. que sans détection, il est impossible d'évaluer les performances de localisation et d'estimation. Les tâches de localisation et d'estimation peuvent être employées pour évaluer et comparer les deux approches de détection proposées : YOLO et SDNet. L'algorithme d'agrégation servant à augmenter la robustesse des détections des tiges de plantes, sa performance est évaluée sur la tâche de localisation de points d'intérêt.

3.1.2 Métriques d'évaluation

Trois types de métriques sont abordées dans cette section. Certaines sont des métriques standards issues de la littérature tandis que d'autres ont été développées pour les besoins spécifiques du travail proposé. Les tâches évaluées sont avant tout des tâches de détection : elles consistent en la détection de plusieurs points d'intérêt, objets (boîtes englobantes) ou structures de plantes. Les métriques de détection, comme celles utilisées pour les bases de données COCO [86] et PASCAL VOC [36] pour la détection d'objets, sont constituées de quatre composants principaux :

- Une métrique de similarité entre deux *instances* (boîte englobante, point d'intérêt ou structure de plante par exemple), servant à mesurer l'adéquation d'une prédiction à une vérité de terrain. La métrique de similarité retourne en général un score entre 0 et 1 où un score de 0 correspond à une dissimilarité absolue (par exemple une

prédiction à l'autre bout de l'image par rapport à la vérité de terrain associée) et un score de 1 correspond à une similarité parfaite (par exemple, la boîte englobante prédite est confondue avec la vérité de terrain). En général, la métrique de similarité ne prend pas en compte la classe de l'instance en compte. Soit l'adéquation du label est évaluée plus tard, soit elle est traitée dès le début en séparant les prédictions et vérité de terrain par classe (évaluation indépendante par classe).

- Un algorithme calculant l'association optimale des prédictions d'une image avec les vérités de terrain, au sens de la mesure de similarité précédente. Cet algorithme classe les prédictions en trois groupes : les vrais positifs (TP), les faux positifs (FP) et les faux négatifs (FN, les oublis). Par exemple, s'il y a une double détection d'un objet (deux boîtes englobantes pour la même vérité de terrain) alors l'une d'entre elles est un faux positif.

- Une métrique quantifiant la pertinence de cette association. Cela peut être une métrique statistique comme le Rappel ou la Précision (définis plus loin) ou des métriques plus complexes comme l'aire sous la courbe Rappel-Précision.

- Une méthode de synthèse des résultats. Les résultats de chaque image sont compilés pour générer un indicateur synthétique pour l'ensemble de la base de données et l'ensemble des classes d'instances. Le plus souvent, la moyenne est employée.

Additionnellement à ces trois métriques, la vitesse d'exécution des algorithmes est utilisée. Celle-ci est couramment mesurée par la latence (temps en seconde) ou par le nombre d'images par seconde (IPS) qu'il est possible d'atteindre sur un support de calcul donné.

3.1.2.1 Métriques de détection d'objets

Le premier type de métrique est constitué des métriques classiques de détection d'objets. Celles-ci sont uniquement employées dans ce travail pour superviser l'apprentissage de la famille de réseaux de détection d'objets YOLO et comparer leur performance. D'autres métriques plus spécifiques aux tâches de localisation et d'estimation sont employées dans la suite pour comparer les performances de l'approche YOLO à SDNet. Les métriques de détection d'objets les plus courantes sont celles proposées pour les bases de données COCO [86] et PASCAL VOC [36]. Celles-ci regroupent en fait une douzaine de métriques similaires dans leur fonctionnement et divergeant principalement sur la "sévérité" de l'évaluation et certains choix d'implémentation. Un état de l'art exhaustif est détaillé dans [105].

Les autres métriques développées dans la suite de la section s'inspirent fortement des métriques des bases de données COCO et PASCAL VOC afin de les adapter aux tâches de localisation et d'estimation du nombre de feuilles, les méthodes de détection d'objets ne sont donc pas approfondies ici. Comme le souligne l'article [59] qui a évalué de nombreux

réseaux de détection d'objets, certaines de ces métriques sont presque parfaitement corrélées ce qui les rend redondantes. Les métriques utilisées dans ce manuscrit se limitent donc aux plus pertinentes et aux plus couramment employées dans la littérature : AP_{50}, AP_{75}, $AP_{[50:95]}$ (noté AP), AR_{100} et aIoU. AP signifie "Average Precision", AR signifie "Average Recall" et aIoU signifie "Average Intersection over Union". L'indice des métriques désigne le seuil d'IoU (*Intersection over Union* qui est une métrique de similarité entre boîtes englobantes décrite plus loin) et l'$AP_{[50:95]}$ est une moyenne calculée pour plusieurs seuils d'IoU.

3.1.2.2 Métriques de localisation de points d'intérêt

Le deuxième type est un ensemble de métriques permettant d'évaluer la performance en localisation de points d'intérêt. Celles-ci sont basées sur le fonctionnement des métriques de PASCAL VOC et COCO et reprennent donc les composants vus plus haut :

- Une métrique de dissimilarité entre deux points d'intérêt.
- Une méthode d'association optimale entre prédictions de points d'intérêt et vérités de terrain d'une image.
- Une métrique quantifiant la pertinence de l'association.
- Une méthode de synthèse des résultats servant à évaluer l'ensemble de la base de données.

Pour une image I_n, cette métrique évalue la qualité des prédictions de points d'intérêt $\hat{D}_n$ définis comme :

$$\hat{D}_n = \left\{ (\hat{p}_{n,i}, \hat{c}_{n,i}, \hat{l}_{n,i}) \right\}_{i \in 1 \ldots \hat{N}_n}, \tag{3.1}$$

où $\hat{p}_{n,i} = (\hat{x}_{n,i}, \hat{y}_{n,i})$ est la position d'un point d'intérêt, $\hat{c}_{n,i}$ la classe (tige de maïs, tige de haricot, feuille), $\hat{l}_{n,i}$ le score de confiance, et $\hat{N}_n$ le nombre de points prédits. Les vérités de terrain D_n de cette image sont définies comme :

$$D_n = \{(p_{n,j}, c_{n,j})\}_{j \in 1 \ldots N_n}, \tag{3.2}$$

où N_n est le nombre de vérités de terrain et les autres paramètres sont analogues à ceux de $\hat{D}_n$. Cette métrique évalue la distance euclidienne (notée "distance") entre une prédiction $\hat{D}_{n,i}$ et une vérité de terrain $D_{n,j}$:

$$\text{distance}(\hat{D}_{n,i}, D_{n,j}) = \sqrt{(\hat{x}_{n,i} - x_{n,j})^2 + (\hat{y}_{n,i} - y_{n,j})^2}. \tag{3.3}$$

Il s'agit de la mesure de (dis)similarité.

L'algorithme d'association est ensuite pratiquement identique à celui présenté dans la section 2.4.2.3 pour l'algorithme d'agrégation de points d'intérêt, l'exception étant

que les prédictions et vérités de terrain sont d'abord filtrées par classe, puis évaluées indépendamment. Ainsi, pour une classe c donnée, l'algorithme d'association procède comme suit :

- Les prédictions $\hat{D}_{n,c}$ de classe c sont triées par ordre décroissant de score de confiance. Ceci permet à l'algorithme d'association de traiter en priorité les prédictions les plus fiables, qui seront donc associées en premier. Ces prédictions triées sont notées $\hat{D}'_{n,c} = \left\{ \hat{D}'_{n,c,j} \,|\, \hat{l}'_{n,c,j} > \hat{l}'_{n,c,j+1} > \right\}_{j \in 1 \ldots \hat{N}_n}$.

- Ensuite, pour chaque prédiction, la vérité de terrain la plus proche au sens de la mesure de dissimilarité est sélectionnée. Cette prédiction est considérée comme un vrai positif (TP) si les conditions suivantes sont réunies : (i) la vérité de terrain n'est pas déjà associée à une autre détection et (ii) la distance entre la prédiction et la vérité de terrain est inférieure à un seuil t_a, c'est-à-dire que la prédiction est jugée suffisamment proche de la vérité de terrain. Dans le cas contraire, la prédiction est considérée comme un faux positif (FP). Puisque les vérités de terrain et les prédictions sont filtrées par classe, la classification du type de point d'intérêt est implicitement incluse dans ces conditions, i.e. que pour être considérée TP une prédiction doit être de la même classe que la vérité de terrain.

- Finalement, toutes les vérités de terrain qui n'ont pas été associées à une prédiction par ce processus sont considérées comme des oublis de détection, c'est-à-dire des faux négatifs (FN).

Pour chaque image et pour chaque classe, on obtient donc une évaluation du nombre de TP, FP et FN. Pour obtenir la synthèse sur la base de données d'images, ces résultats sont simplement sommés. On obtient donc le total de TP, FP et FN toutes classes confondues et pour toutes les images.

Pour finir, plusieurs métriques synthétiques sont calculées sur ces résultats. La *précision* et le *rappel* :

$$\begin{aligned} \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{rappel} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \end{aligned} \tag{3.4}$$

le *Score F1* (moyenne harmonique du rappel et de la précision) :

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{3.5}$$

et la moyenne des erreurs absolues (MAE) en localisation MAE_{loc}, définie comme la

distance moyenne des prédictions jugées TP $\hat{D}_n^{\mathrm{TP}}$:

$$\mathrm{MAE}_{\mathrm{loc}} = \sum_{\hat{D}_{n,i}^{\mathrm{TP}} \in \hat{D}_n^{\mathrm{TP}}} \mathrm{sim}(\hat{D}_{n,i}^{\mathrm{TP}}, D_{n,i}), \tag{3.6}$$

où $D_{n,i}$ est la vérité de terrain associée à $\hat{D}_{n,i}^{\mathrm{TP}}$. La précision et le rappel mesurent respectivement la propension du détecteur à générer des faux positifs et des oublis (faux négatif). Par exemple, un score de rappel bas indique que le détecteur de points d'intérêt détecte trop peu de vérités de terrain, soit parce qu'il ne détecte rien, soit parce qu'il se trompe de classe, parce qu'il localise systématiquement à côté de la vérité de terrain. Au contraire, un score de précision bas indique que le détecteur génère trop de faux positifs, i.e. des points d'intérêt qui ne correspondent à aucune vérité de terrain. La précision mesure donc le taux de détections invalides parmi les prédictions et le rappel mesure le taux d'oublis de détections parmi les vérités de terrain. Le Score F1 est simplement une mesure synthétique de ces deux quantités. Le $\mathrm{MAE}_{\mathrm{loc}}$ quant à lui évalue l'erreur moyenne de localisation, i.e. l'erreur à laquelle on peut s'attendre sur la position d'une prédiction. Cette erreur est quantifiée en mètres.

Le seuil de distance minimal t_a pour qu'une détection soit jugée assez proche d'une vérité de terrain est fixé à une valeur de 2 cm, i.e. la prédiction doit être dans un rayon de 2 centimètres autour de la vérité de terrain pour être considérée valide. Cette valeur est fixée expérimentalement et correspond à une valeur suffisante pour réaliser du désherbage de précision avec le bloc-outil BIPBIP[1].

3.1.2.3 Métriques d'estimation du nombre de feuilles

Cette métrique est très similaire à la métrique de localisation de points d'intérêt dans son fonctionnement. Elle ne considère pas des points d'intérêt mais des détections de plante entière, et elle évalue la capacité d'estimation du nombre de feuilles. La différence avec la métrique précédente se situe au niveau des conditions nécessaires pour qu'une prédiction soit considérée comme une vérité de terrain. Ici, non seulement il faut que la prédiction ait la même classe que la vérité de terrain mais il faut aussi que le nombre de feuilles prédit soit identique à la vérité de terrain. Par exemple, une prédiction estimant un nombre de feuilles inférieur à celui de la vérité de terrain sera considérée comme un faux positif.

Une autre distinction est liée à la différence de fonctionnement entre les deux détecteurs proposés. YOLO détecte les plantes (et le nombre de feuilles) comme des boîtes englobantes alors que SDNet prédit des graphes étoilés décrivant la structure de la plante. La métrique de similarité entre prédictions et vérités de terrain employée est donc nécessairement

[1]En pratique, la valeur de ce seuil dépend de l'application et doit être adaptée en fonction de la tâche visée. Le choix de la valeur précise est plutôt subjectif, tout comme le seuil d'IoU de l'AP, analogue au seuil de distance t_a. Le choix d'un seuil plus restrictif (i.e. plus petit) produira des scores moins élevés mais les performances relatives devraient rester les mêmes [59].

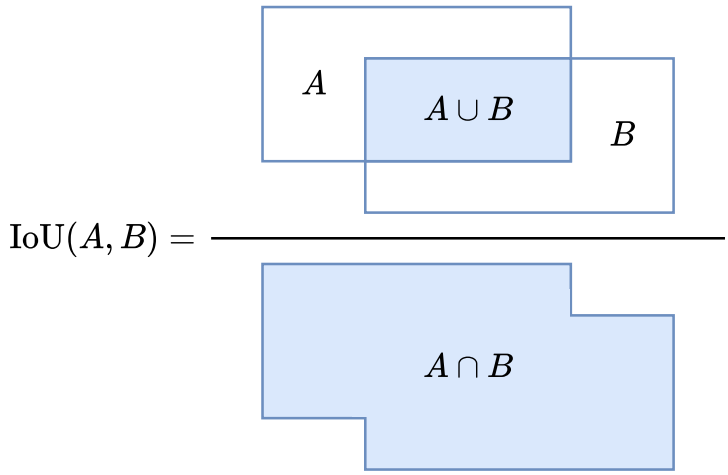


FIGURE 3.3 : Illustration de la mesure de similarité IoU (*Intersection over Union*) entre deux boîtes englobantes. L'IoU est défini comme l'aire de l'intersection divisée par l'aire de l'union des boîtes englobantes. Plus il est élevé, plus les boîtes sont similaires.

différente pour les deux propositions. SDNet utilise l'ancre de la plante, donc la mesure de similarité est la même que celle présentée pour la métrique précédente (distance entre l'ancre prédite et l'ancre de la vérité de terrain). Pour YOLO, la mesure de similarité IoU (*Intersection over Union*) est utilisée pour mesurer la similarité entre deux boîtes englobantes[2]. La figure 3.3 illustre cette mesure. Elle est définie par l'aire de l'intersection divisée par l'aire de l'union des boîtes englobantes. Une valeur nulle indique une forte dissimilarité (les boîtes ne s'intersectent pas) et la plus haute valeur, égale à 1, indique une similarité parfaite (les boîtes sont exactement superposées).

Ces différences mises à part, les métriques synthétiques *précision*, *rappel* et *score F1* sont définies de la même manière que précédemment. Cependant, la métrique de $\mathrm{MAE}_{\mathrm{loc}}$ est remplacée par une métrique $\mathrm{MAE}_{\mathrm{count}}$ évaluant l'erreur moyenne absolue sur le nombre de feuilles prédites. Elle est définie pour les détections $\hat{D}_n^{\mathrm{TFP_c}}$ qui sont soit des vrais positifs, soit des faux positifs causés par une erreur d'estimation du nombre de feuilles[3] par :

$$\mathrm{MAE}_{\mathrm{count}} = \sum_{\hat{D}_{n,i}^{\mathrm{TFP_c}} \in \hat{D}_n^{\mathrm{TFP_c}}} \left| \mathrm{nb\_leaf}(\hat{D}_{n,i}^{\mathrm{TFP_c}}) - \mathrm{nb\_leaf}(D_{n,i}) \right|, \tag{3.7}$$

où $D_{n,i}$ est la vérité de terrain correspondant à $\hat{D}_{n,i}^{\mathrm{TFP_c}}$ (la vérité de terrain associée à cette détection par le processus d'association optimal des vérités de terrain aux prédictions) et `nb_leaf` la fonction retournant le nombre de feuilles d'une prédiction ou vérité de terrain. Le $\mathrm{MAE}_{\mathrm{count}}$ évalue donc l'erreur moyenne commise sur la prédiction du nombre de feuilles.

[2]Un seuil sur l'IoU de 50 % est utilisé, le plus permissif, pour juger si une prédiction est TP ou FP.

[3]Ce sont les seuls cas où l'erreur sur le nombre de feuilles est définie, i.e. une prédiction bien localisée et ayant la bonne classe.

3.2 Bases de données et entraînements

Cette section présente d'abord les bases de données d'images et de séquences d'images qui ont été acquises pour entraîner les algorithmes d'apprentissage profond et pour valider les performances. Ensuite, les détails des entraînements des réseaux sur ces données sont présentés.

3.2.1 Base de données d'images et annotations

Afin d'entraîner les réseaux d'apprentissage profond et de valider les performances des algorithmes, trois bases de données d'images et de séquences d'images ont été acquises et annotées. Elles se distinguent par le type d'annotation qu'elles contiennent et donc par le type d'algorithme qu'elles permettent d'entraîner et par le type de tâche qu'elles permettent d'évaluer :

- DB_1 : c'est une base de données d'images annotées pour la détection d'objets, elle est donc spécifique à l'approche YOLO. C'est la base la plus ancienne et aussi la plus complète : c'est la seule qui contient des annotations de poireaux, une espèce ajoutée en dernier dans la base. Elle n'est utilisée que pour les recherches exhaustives de paramètres optimaux de YOLO.

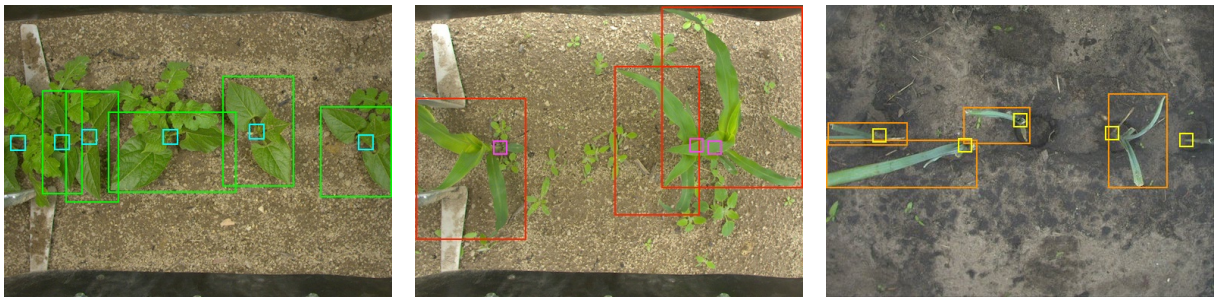
- DB_2 : c'est une base de données d'images annotées pour entraîner et valider les deux approches neuronales et pour évaluer les deux tâches sélectionnées : la localisation d'organes et l'estimation du nombre de feuilles. C'est la base la plus récente et la moins complète. Plus réduite que DB_1, elle contient seulement des annotations de haricot et de maïs et contient moins d'images. C'est aussi la base la plus polyvalente car elle permet d'évaluer les deux tâches et d'entraîner les deux réseaux de neurones. Elle est rétro-compatible avec DB_1.

- DB_3 : c'est une base de données de séquences d'images créée pour évaluer l'algorithme d'agrégation sur la tâche de localisation de tiges. Elle ne contient que des annotations de maïs et de haricot.

Les deux premières bases de données ont fait l'objet d'une publication [78] et sont disponibles en source ouverte[4].

Toutes les images et séquences sont acquises avec la caméra du bloc-outil BIPBIP dans les conditions décrites dans la section 1.1. Un module d'acquisition léger et déplaçable manuellement, nommé Pollux, a aussi été employé pour réaliser des acquisitions dans les mêmes conditions que BIPBIP et pour partager le travail entre les membres du projet. Les images et séquences sont acquises sur des parcelles chez des partenaires (CTIFL, Fermes Larrère) ou lors d'évaluations de terrain sur le site INRAE de Montoldre.

[4] `https://data.mendeley.com/datasets/d7kbzjr83k/1`



(a) Haricot. (b) Maïs. (c) Poireau.

FIGURE 3.4 : Illustration d'images annotées de DB_1. a et b sont des images de haricot et de maïs acquises avec le bloc-outil BIPBIP. c est une image de poireau acquise avec le module Pollux. En superposition sont représentées les annotations : boîtes englobantes de plantes entières et boîtes de taille fixes centrées sur les bases de tiges, i.e. le point d'ancrage dans le sol.

Trois types de cultures sont pris en charge : le maïs, le haricot et le poireau. La figure 3.4 illustre des échantillons d'images tirés de la base de données DB_1. Le poireau est la dernière plante ajoutée aux bases de données, il est donc sous-représenté et n'est pas présent dans toutes les bases. Les images (ainsi que celles des séquences) ont une définition de 3 millions de pixels[5]. L'annexe D illustre davantage d'échantillons.

3.2.1.1 Base de données d'images DB_1

La première base DB_1 est une base de données d'images annotées couvrant trois variétés de plantes : le maïs, le haricot et le poireau. Toutes les images sont annotées pour la détection d'objets et deux types d'objets sont annotés : la plante entière et la position estimée de la base de la tige de la plante. La figure 3.4 illustre un échantillon d'image annotée pour chaque variété de plantes. La boîte englobante de tige est carrée, de taille fixe et centrée sur la position estimée de la base de la tige afin d'entraîner les réseaux YOLO, comme présenté dans la section 2.2. Le tableau 3.1 comptabilise le nombre d'images et d'annotations ventilé par variété de plante. Le poireau apparaît sous-représenté (seulement 227 images) par rapport au maïs (plus de 1 000 images) et au haricot (748 images). Il y a néanmoins plus de haricots par image car le schéma de semis est plus serré que pour le maïs (voir section 1.1), donc il y a plus d'annotations de haricot que de maïs malgré un nombre d'images inférieur. Des images vides (sol nul ou uniquement des adventices) sont aussi intégrées à cette base de données afin d'éviter le sur-apprentissage. Le tableau 3.1 montre aussi qu'il n'y a pas forcément le même nombre d'annotations de plantes entières que de tiges. Ceci est dû aux plantes en bordure d'image dont on ne voit que la base de la tige à cause de la forte parallaxe. Dans ces cas, seule la base de la tige est annotée. Un exemple est illustré sur l'image 3.4c tout à droite ou sur l'image 3.4a tout à gauche.

[5]Quelques anciennes images ont une définition de 5 millions de pixels et un rapport hauteur/largeur légèrement différent.

Classe	Images	Plantes	Tiges
Maïs	1 034	2 095	2 133
Haricot	748	2 820	2 824
Poireau	227	1 111	1 191
Vide	135	0	0
Total	2 144	6 026	6 148

TABLEAU 3.1 : Statistiques de la base de données DB_1 avec le nombre d'images, d'annotations de plantes entières et le nombre d'annotations de bases de tige. La classe "Vide" représente les images sans annotation.

Cette base de données d'images DB_1 n'est utilisée que pour la comparaison des architectures YOLO entre elles et pour la recherche des hyper-paramètres optimaux de YOLO présentés dans la section 3.3.1. L'annotation des données est réalisée avec le logiciel open source LabelImg[6], le format d'annotation est un fichier XML par image (format PASCAL VOC [36]).

3.2.1.2 Base de données d'images DB_2

La seconde base DB_2 est une base de données d'images annotées couvrant deux variétés de plantes : le maïs et le haricot. Les images sont annotées avec les structures de plantes et les boîtes englobantes de plantes entières. La structure d'une plante est définie par la position estimée de la base de la tige et des feuilles de la plante (pointes de feuilles). Pour plus de détail, voir la section 2.3. La figure 3.5 illustre deux annotations de haricot (ligne du haut) et deux annotations de maïs (ligne du bas), les points verts correspondent aux tiges, les points rouges aux feuilles et la boîte englobante est en bleu. La variété de la plante est indiquée par un label à côté de la tige de la plante. Des échantillons d'images supplémentaires sont illustrés en annexe D.

Le nombre d'ancres de cette base de données est de deux : l'ancre de la base de tige de maïs et celle du haricot. Il n'y a qu'une seule partie annotée, commune aux deux espèces de plante : la (pointe de) feuille. La structure de la plante permet en réalité de différencier les deux types de feuilles (feuilles de maïs et feuilles de haricot) puisque les points d'intérêt de feuille sont nécessairement associés à une ancre (maïs ou haricot). En revanche, le type de la feuille n'est pas nécessaire pour réaliser les deux tâches évaluées puisque cette information n'est pas nécessaire pour le comptage du nombre de feuilles ou pour le désherbage de précision visé par BIPBIP qui ne requiert que la position de la base de la tige. Afin de simplifier les expérimentations, les deux types de feuilles ne sont pas différenciés dans la suite : le point d'intérêt de feuille est considéré comme *agnostique* de l'espèce de la plante. Une expérimentation différenciant les types de feuilles est néanmoins présentée en annexe B.

Cette base de données permet d'entraîner les deux réseaux de neurones profonds sur

[6] `https://github.com/tzutalin/labelImg`

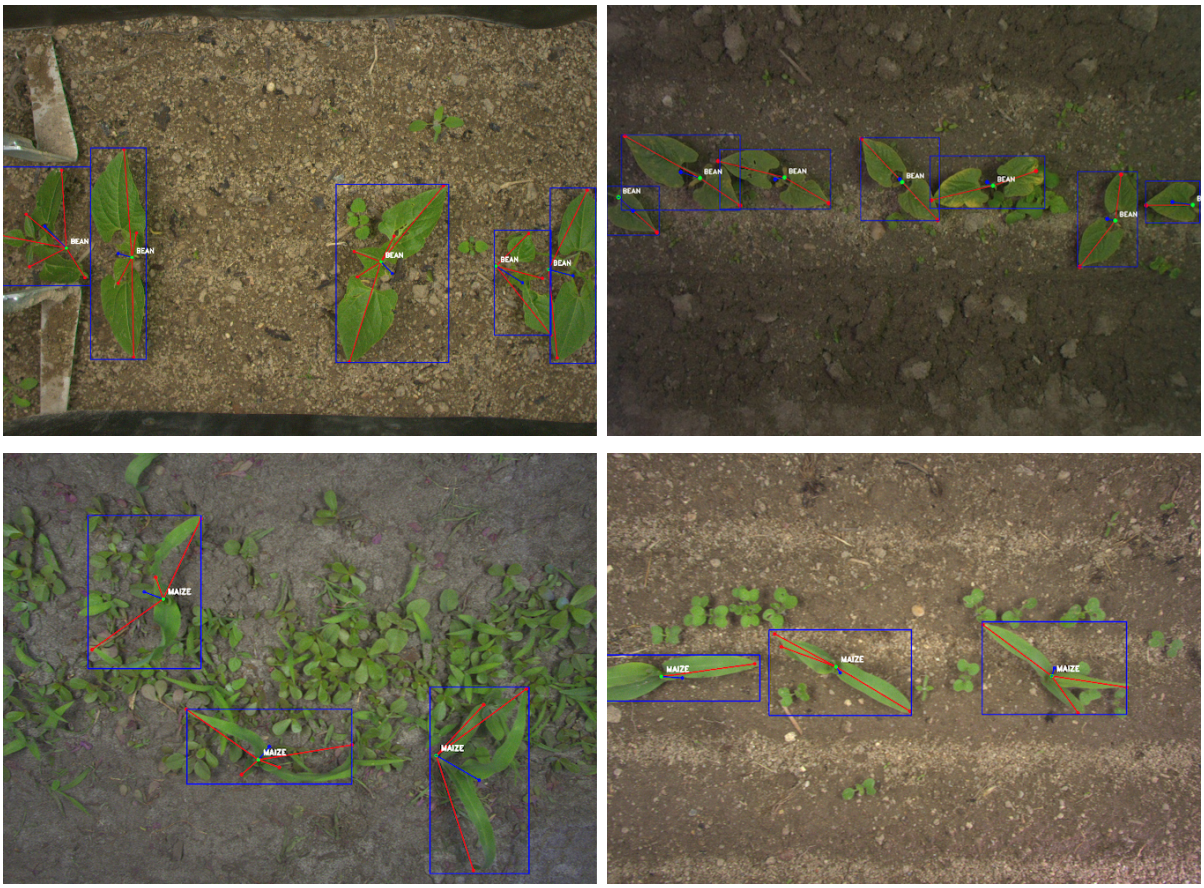


FIGURE 3.5 : Illustration de deux images annotées de haricot (première ligne) et de maïs (ligne du bas) de la base de données DB_2. La structure de plante est annotée avec le point d'intérêt de la tige (en vert) et les pointes de feuilles (en rouge), les boîtes englobantes de plante sont en bleu. Le nombre de feuilles correspond au nombre de parties (en rouge).

les deux tâches évaluées : la localisation d'organes et l'estimation du nombre de feuilles. En effet, chaque annotation de plante (structure de plante et boîte englobante) fournit toutes les informations nécessaires pour les deux algorithmes : localisation des organes, nombre de feuilles, et relations entre les organes. Les points d'intérêt n'étant pas dans le cadre de l'image ne sont pas annotés. Par exemple, sur la figure 3.5, dans l'image en haut à droite, les deux haricots sur les bords gauche et droit de l'image ne sont annotés qu'avec une seule feuille bien qu'ils en aient probablement une seconde.

Le tableau 3.2 comptabilise le nombre d'images et de points d'intérêt annotés ventilé par variété de plante (maïs ou haricot). Une plante n'est pas annotée si on ne voit pas sa tige dans l'image, il y a donc autant d'annotations de bases de tiges que d'annotations de plantes entières (boîte englobante). La répartition des annotations par variété est relativement équilibrée, il y a plus d'images de maïs que de haricot (+56 %) mais puisque le schéma de semis est plus serré pour le haricot (voir section 1.1) il y a plus d'annotations de haricot que de maïs (respectivement 1 302 et 1 067, soit +22 %). Les plantes étant à un stade précoce (cf. section 1.1), le nombre de feuilles par plante est faible et d'environ 2 feuilles par plantes. Une étude plus approfondie de la répartition du nombre de feuilles par

Classe	Images	Tiges	Feuilles
Maïs	497	1 067	2 647
Haricot	317	1 302	2 834
Vide	60	0	0
Total	874	2 369	5 481

TABLEAU 3.2 : Statistiques de la base de données DB_2 comportant le nombre d'images, d'annotations de tige et de feuilles correspondant à chaque classe. La classe "Vide" regroupe les images sans annotation.

plante est présentée dans l'annexe E. Au total, il y a près de deux fois moins d'images (et d'annotations) que dans la base de données DB_1 car elle a été conçue plus tardivement.

La base de données DB_2 est la plus polyvalente. Elle permet d'entraîner les deux réseaux de neurones et de les évaluer sur les tâches de localisation d'organes et d'estimation du nombre de feuilles (voir sections 3.4.1.1 et 3.4.1.2). Cette base de données est aussi utilisée pour les recherches des paramètres optimaux de SDNet (voir section 3.3.2) et pour comparer les performances de YOLO et SDNet (voir section 3.4.1).

À notre connaissance, il n'y a pas de logiciel d'annotation de structure d'objet. De ce fait un format d'annotation spécifique à la détection de structures en étoiles et un logiciel d'annotation ont été développés. Le format d'annotation sélectionné est basé sur le format de sérialisation JSON, universellement interopérable et simple à utiliser. Le logiciel d'annotation permet d'annoter à la fois la structure en étoile et la boîte englobante de chaque plante d'intérêt. Développé avec Python et OpenCV, il est disponible en open source[7].

3.2.1.3 Base de données de séquences d'images DB_3

La dernière base de données, DB_3, est une base de séquences d'images annotées utilisée pour évaluer les performances de l'algorithme d'agrégation de point d'intérêt de tige. Elle supporte deux variétés de plante : le maïs et le haricot. Seuls les points d'intérêt de tige de plantes sont annotés sur ces images. La figure 3.6 illustre deux extraits de séquence d'images sur le maïs (ligne du haut) et sur le haricot (ligne du bas). Les bases de tiges sont annotées en rouge pour le maïs et en orange pour le haricot et une flèche identifie une plante en particulier afin d'illustrer le mouvement. L'extrait de maïs illustre la forte parallaxe présente dans les séquences d'images : l'aspect de la plante de maïs annoté par la flèche varie fortement d'une image à l'autre. Au début de la séquence, le haut de la plante est masqué puis, au milieu de la séquence, la base de la tige est cachée par les feuilles. La parallaxe est moins visible sur le haricot puisqu'ils sont plus petits. Les deux séquences montrent que la base de la tige est masquée (par des feuilles, une autre plante ou des adventices) la majeure partie du temps. Observer plusieurs points de vue de la

[7] `https://github.com/laclouis5/StructureAnnotator`

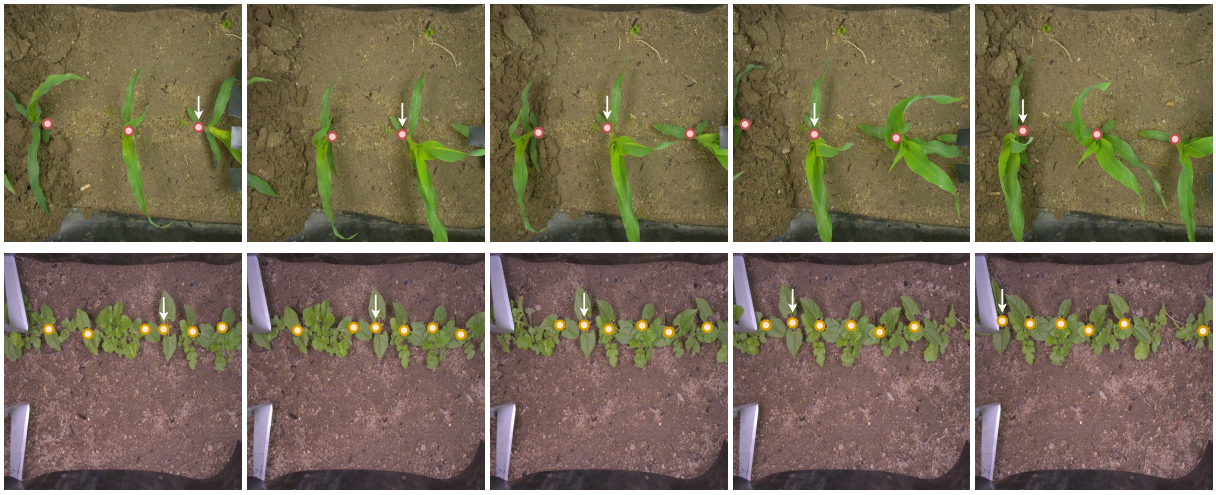


FIGURE 3.6 : Illustration de deux séquences d'images de DB_3 sur le maïs (ligne du haut) et le haricot (ligne du bas). Une base de tige de maïs (en rouge) et de haricot (en orange) sont annotées pour illustrer le déplacement de la caméra. La cadence de prise de vue est volontairement abaissée et les images recadrées pour améliorer la visualisation.

Séquence	Images	Images annotées	Tiges	Fréquence d'annotation
Haricot 1	251	23	94	10
Haricot 2	784	30	169	26
Maïs 1	1 765	43	124	41
Maïs 2	427	8	20	53

TABLEAU 3.3 : Statistiques de la base de données DB_3 sur les quatre séquences présentant le nombre d'images des séquences, le nombre d'images annotées, le nombre de bases de tige de plante annotées et la fréquence d'annotation (nombre d'images de la séquence divisé par le nombre d'images annotées).

plante permet de bénéficier de plus d'opportunités de détecter la base de la tige.

La base de données est constituée de quatre séquences, deux pour le maïs et deux pour le haricot. Le tableau 3.3 comptabilise le nombre d'images total de chaque séquence, le nombre d'images annotées et le nombre de bases de tige annotées. Toutes les images ne sont pas annotées afin de réduire le temps d'annotation et de ne pas annoter plusieurs fois la même plante (environ une toute les 10 à 50 images selon la vitesse d'avancement et la cadence de prise de vue). Chaque plante de chaque séquence est donc annotée qu'une seule fois.

Cette base de données est utilisée pour réaliser la recherche des paramètres optimaux de l'algorithme d'agrégation et évaluer ses performances.

3.2.2 Plans d'entraînement

Cette section présente le détail des entraînements réalisés pour les réseaux de la famille YOLO et pour SDNet sur les bases de données d'entraînement DB_1 et DB_2 introduites précédemment. Certains paramètres de l'entraînement, comme la définition de l'image

d'entrée, ont fait l'objet de recherches exhaustives présentées plus loin. Les valeurs utilisées ici correspondent aux valeurs typiques et servent à illustrer la méthode d'entraînement. Les bases de données sont systématiquement divisées aléatoirement en un jeu d'entraînement comportant 80 % des données et un jeu de validation comportant les 20 % restant. Le jeu de validation sert pour la recherche des hyperparamètres des réseaux et pour l'évaluation des performances. Cette séparation en deux jeux de données, l'un pour l'entraînement et l'autre pour la validation, permet de d'éviter le risque de sur-apprentissage (*overfitting*) des réseaux de neurones. L'absence d'un jeu de données de test, motivé en partie par la faible taille des jeux de données, peut en revanche introduire un biais dans le choix des hyperparamètres. L'annexe F présente des considérations sur la capacité de sur-apprentissage et de généralisation des réseaux entraînés et explique en quoi le biais introduit par l'absence de jeu de test est en pratique limité.

Les réseaux de neurones profonds sont entraînés sur un serveur de calcul tournant sous Ubuntu 20.04 LTS doté d'un CPU à 4 cœurs cadencés à 3,6 GHz et de 32 Go de mémoire RAM. Il est équipé d'un GPU NVIDIA GeForce RTX 2080 SUPER doté de 8 Go de mémoire graphique afin d'accélérer les entraînements.

3.2.2.1 Entraînement de YOLO

Comme présenté dans la section 3.3.1, les réseaux de détection d'objets YOLO peuvent être employés pour de la localisation de points d'intérêt et pour la caractérisation de plantes. Cela requiert néanmoins deux entraînements séparés. Au total, trois configurations d'entraînement sont établies :

- La configuration Y_{hp}, qui permet d'entraîner YOLO sur la base de données DB_1 et de sélectionner les meilleurs hyperparamètres des réseaux, comme la définition des images et le seuil sur la confiance. DB_1 est la base de données la plus complète et permet donc de limiter le sur-apprentissage de ces paramètres.

- La configuration Y_{kp}, qui permet d'entraîner les réseaux YOLO pour la tâche de localisation d'organes de plantes et qui utilise la base de données DB_2. La base de données DB_2 est moins complète, mais permet une comparaison à jeu de données égal avec SDNet. Dans cette configuration, YOLO est entraîné à détecter 3 points d'intérêt : les bases de tiges de maïs et de haricot, et les extrémités de feuilles.

- La configuration Y_{cl}, qui permet d'entraîner YOLO pour la tâche d'estimation du nombre de feuilles de plante. Cette configuration utilise aussi la base de données DB_2 pour autoriser une comparaison équitable avec SDNet. Dans cette configuration, YOLO est entraîné à détecter deux variétés de plantes (maïs et haricot) et à estimer un nombre de feuilles entre 0 et 8 (le maximum présent dans la base de données).

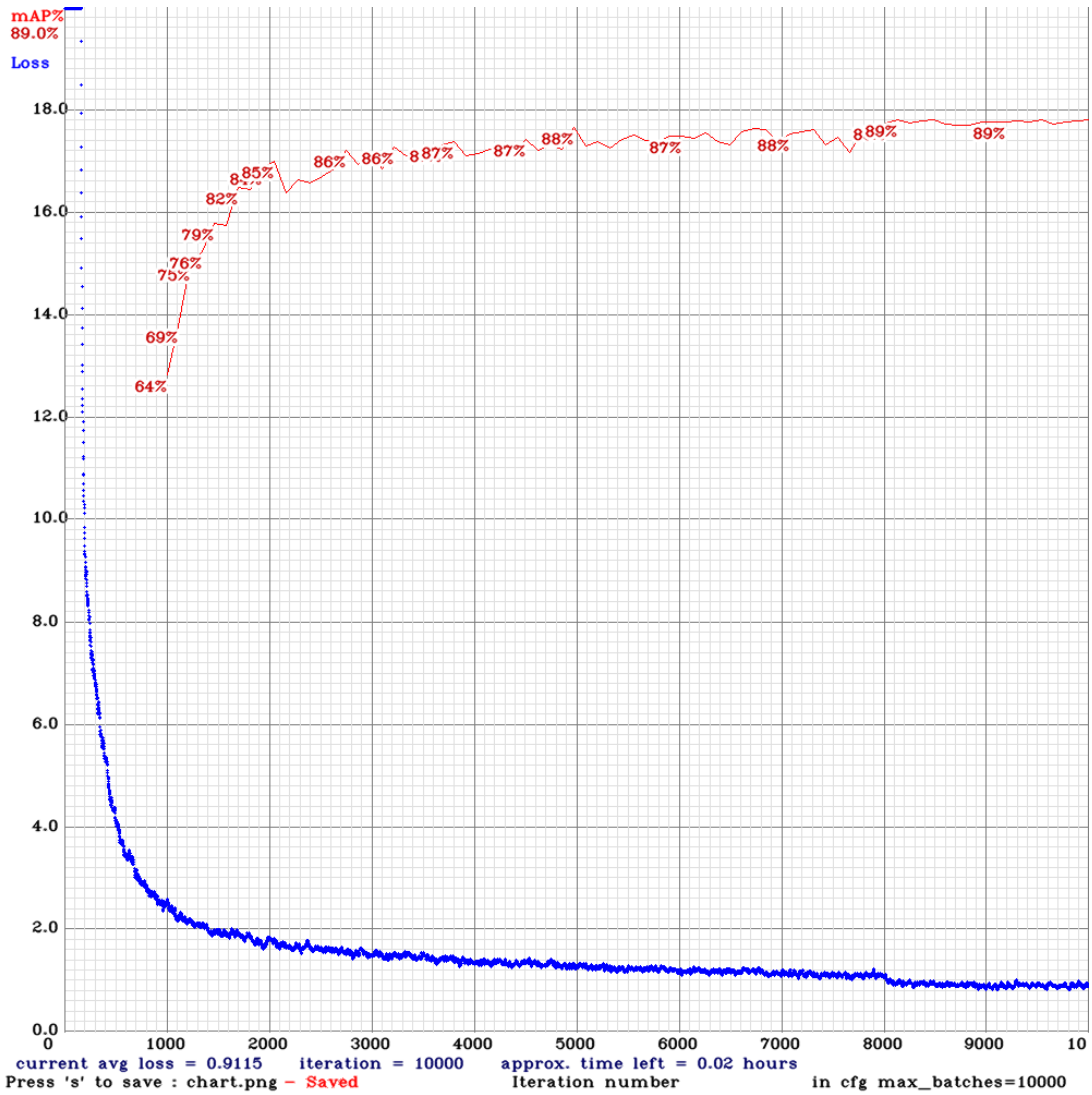


FIGURE 3.7 : Évolution de la valeur de la fonction de coût (en bleu) au cours de l'entraînement de YOLOv4 Tiny et de la valeur de la métrique mAP calculée sur le jeu de données de validation (en rouge). L'entraînement est réalisé sur la base de données DB_1.

Pour l'entraînement, le framework Darknet[8] présenté dans [13] est utilisé. Le framework inclut des augmentations de données faisant partie intégrante de la méthode optimale d'entraînement des réseaux, les paramètres par défaut sont donc utilisés (il y a notamment des changements de couleur, de définition d'image, des rotations et d'autres méthodes plus avancées comme Mosaïc [13] qui mélange quatre images en entrée du réseau). La taille de batch est de 64 images et le nombre d'itérations (une itération correspond à une batch) est de 10 000. L'apprentissage par transfert [127] (*Transfer Learning*) est employé en initialisant les poids synaptiques du réseau avec les poids du même réseau entraîné sur la base de données COCO [86].

La figure 3.7 illustre les courbes d'entraînement typiques de l'apprentissage d'un réseau de la famille YOLO (ici YOLOv4 Tiny) réalisé sur la base de données DB_1. La courbe de

[8] `https://github.com/AlexeyAB/darknet/`

coût est présentée en bleu (calculée sur le jeu d'entraînement) et la courbe de validation utilisant le critère mAP (*Mean Average Precision* [36]) est présentée en rouge (calculée sur le jeu de validation). Elle montre une convergence stable qui sature rapidement au bout de quelques milliers d'itérations. En pratique, l'entraînement peut être stoppé dès que la métrique d'évaluation n'augmente plus de façon significative (autour de 5 000 itérations[9]).

La durée de l'entraînement varie en fonction de la profondeur du réseau (cf. section 2.2.3 pour l'explication et la section 3.3.1.1 pour une comparaison de la vitesse d'inférence). Elle varie de 2 heures pour YOLOv4 Tiny (la variante la plus rapide de la famille YOLO) à 7 heures pour YOLOv4 (la variante la plus précise). La définition des images en entrée du réseau influence aussi la durée d'entraînement et la vitesse d'inférence (cf. section 3.3.1.4).

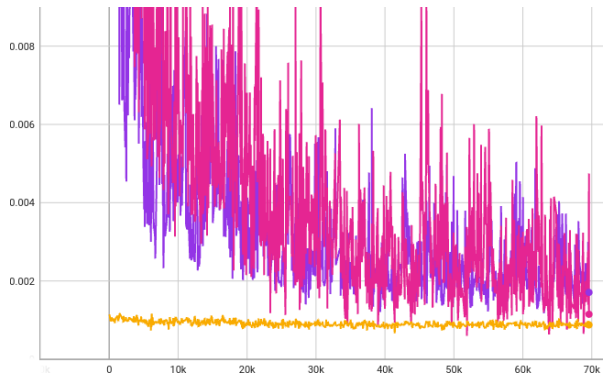
3.2.2.2 Entraînement de SDNet

Le réseau de neurones SDNet est conçu pour détecter des structures de plantes permettant à la fois de localiser les organes et de compter le nombre de feuilles. Il ne requiert donc qu'un seul entraînement. La base de données de structures de plante DB_2 est utilisée pour l'entraînement, la recherche des hyperparamètres optimaux et pour la comparaison avec les réseaux YOLO. Entraîné sur cette base de données, SDNet est capable de localiser trois types de points d'intérêt (bases de tiges de maïs, de haricot, et les feuilles) et d'estimer un nombre arbitraire de feuilles. En réalité, SDNet est capable de distinguer implicitement les deux catégories de feuilles (feuilles de maïs ou de haricot) sans avoir à définir plusieurs types de parties, contrairement à YOLO. En effet, la structure en graphe étoilé implique qu'une partie est nécessairement associée à une ancre, ce qui permet de "typer" la partie avec le label de l'ancre (maïs ou haricot dans notre cas). En revanche, cette information n'est pas nécessaire pour estimer le nombre de feuilles ou pour le désherbage de précision qui ne requiert que la position de la base de la tige. Pour simplifier les expérimentations réalisées dans la suite, la partie de feuille est considérée comme agnostique de l'espèce de la plante dans la suite. L'annexe B détaille une expérimentation dans le cas non-agnostique.

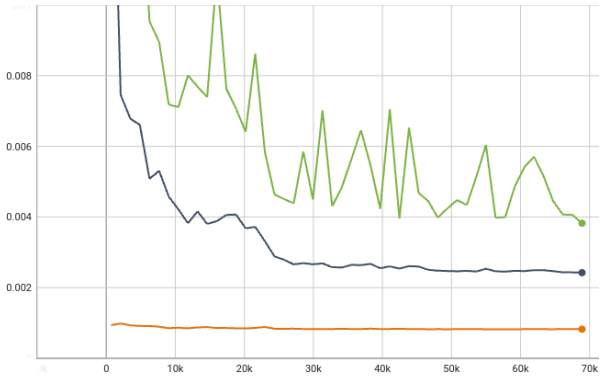
Le code d'entraînement est développé en PyTorch et disponible en Open Source[10]. Plusieurs augmentations de données sont implémentées (tirage aléatoire uniforme) : miroir horizontal et vertical, modifications colorimétriques (luminosité ±25 %, contraste ±25 %, saturation ±15 % et teinte ±5 %) et changement de définition (échelle) entre ±25% de la définition d'origine. La taille de batch est fixée à 8 et la durée d'entraînement est de 100 époques (soit d'environ 8 700 itérations). Le taux d'apprentissage (*Learning Rate*) est fixé à 0,001 et est divisé par 10 toutes les 33 époques et l'optimiseur Adam [70] est employé. Pour finir, les poids de la fonction de coût (voir équation 2.17) sont fixés à

[9] Le palier dans la courbe de coût visible à l'itération 8 000 est dû à une méthode d'entraînement visant à réduire le taux d'apprentissage à la fin de l'entraînement pour améliorer les performances (le mAP augmente légèrement). Cette augmentation finale n'est donc pas due à la durée d'entraînement et serait présente vers l'itération 4 000 si l'entraînement était réduit à 5 000 itérations.

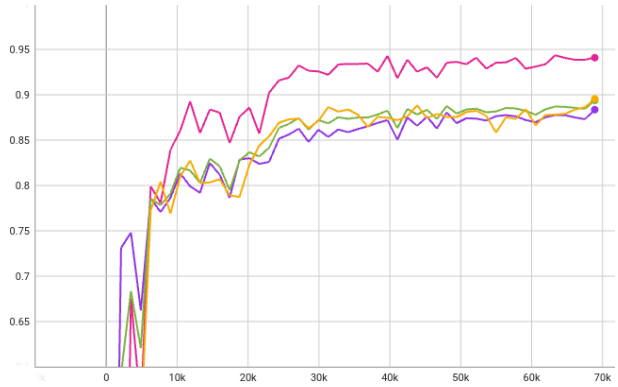
[10] `https://github.com/laclouis5/StructureDetector`



(a) Évolution de la valeur du coût à l'apprentissage des cartes de chaleur (rose), des indices de regroupement (violet) et des vecteurs de compensation (jaune).



(b) Coût en fonction du temps (jeu de validation) des cartes de chaleur (vert), des indices de regroupement (gris) et des vecteurs de compensation (orange).



(c) Score F1 en fonction du temps (jeu de validation) pour le point d'intérêt de tige de maïs (rose), de tige de haricot (jaune), de feuille (violet) et total (vert).

FIGURE 3.8 : Courbes de l'évolution du coût lors de l'apprentissage (a) et de la validation (b) et courbes de l'évolution de la métrique d'évaluation Score F1 sur la tâche détection d'organes (c). L'entraînement est réalisé sur le jeu d'entraînement de la base DB_2. L'axe des abscisses correspond au nombre d'images entraînées.

$\lambda_y = 1,0$, $\lambda_o = 0,001$ et $\lambda_e = 0,001$ afin que les trois composantes de la fonction de coût soient approximativement de même amplitude en fin de convergence. L'apprentissage par transfert [127] est utilisé en initialisant les poids synaptiques de la colonne avec les poids du réseau entraîné sur la base de données COCO [86]. La durée d'entraînement est environ d'une heure.

Le framework PyTorch est plus flexible que Darknet et permet de surveiller l'avancement des entraînements et de journaliser les métriques plus facilement. Les figures 3.8a et 3.8b illustrent l'évolution de la valeur de la fonction de coût éclatée par composante (cartes de chaleur, indices de regroupement et vecteurs de compensation de décimation) respectivement calculée sur le jeu d'entraînement et sur le jeu de validation de la base DB_2 lors d'un entraînement typique de SDNet. L'amplitude de la courbe des vecteurs

de compensation de décimation est plus faible que les deux termes et évolue peu, ce qui peut signifier que ces vecteurs apprennent peu. La section 3.3.2.3 analyse l'impact des vecteurs de décimation sur la performance. L'évolution de la fonction de coût des cartes de chaleurs est plus fluctuante que les deux autres fonctions de coût car elle est probablement plus "sollicitée" lors de l'apprentissage. L'apprentissage des cartes de chaleur est en effet fondamental pour l'apprentissage : l'apprentissage des indices de regroupement et des vecteurs de compensation de la décimation impliquent que les points d'intérêts soient correctement détectés et localisés. La figure 3.8c présente l'évolution du score F1 de la tâche de localisation d'organes calculée sur la base de validation et ventilée par type d'organe (tige de maïs, tige de haricot et feuille). Ces courbes, et notamment la métrique de validation, montrent que l'apprentissage converge rapidement.

La figure 3.9 illustre les tenseurs $\hat{Y}$, $\hat{E}$ et $\hat{E}$ prédits par SDNet sur un exemple d'image de maïs (sous-figure a). Les cartes de chaleur prédites de l'ancre du maïs et de la partie de feuille sont illustrées dans les sous-figures b et c. Les courbes gaussiennes aux positions des bases de tiges et de parties se distinguent clairement, même lorsque les points d'intérêt sont proches comme pour certaines feuilles. Le champ vectoriel d'indices de regroupement (d) montre que les vecteurs pointent vers les ancres dans une large zone autour des points d'intérêt de parties. Le champ vectoriel de compensation (e) montre des vecteurs de faible amplitude, ce qui est attendu puisqu'ils correspondent à la partie fractionnaire de la position des points d'intérêt.

3.3 Paramétrage optimal des algorithmes

Cette section présente plusieurs recherches exhaustives de paramètres des réseaux de neurones et algorithmes. Ces recherches permettent d'une part d'optimiser les performances et d'autre part d'analyser l'influence des paramètres sur la performance et la robustesse. Les recherches exhaustives sont gourmandes en ressources de calcul puisqu'elles nécessitent un nombre d'exécutions conséquent des algorithmes. Elles le sont d'autant plus lorsqu'elles requièrent un apprentissage complet d'un réseau de neurones. Les expérimentations présentées se limitent donc aux paramètres principaux des algorithmes, et ceux-ci sont majoritairement analysés de manière indépendante les uns des autres.

Concernant YOLO, plusieurs architectures plus ou moins profondes ont été évaluées ainsi que plusieurs définitions d'entrée du réseau. La taille de la boîte englobante de tige est aussi analysée et une recherche exhaustive sur la valeur du seuil de confiance optimal est reproduite. Pour SDNet, une recherche exhaustive réalisée conjointement sur le seuil de confiance t_c et le seuil de distance du décodeur t_d est présentée ainsi qu'une recherche exhaustive sur la taille de noyau gaussien de l'encodeur et qu'une analyse du gain de performance apporté par les vecteurs de compensation de la décimation. Enfin, pour l'algorithme d'agrégation temporelle, une recherche exhaustive conjointe sur le

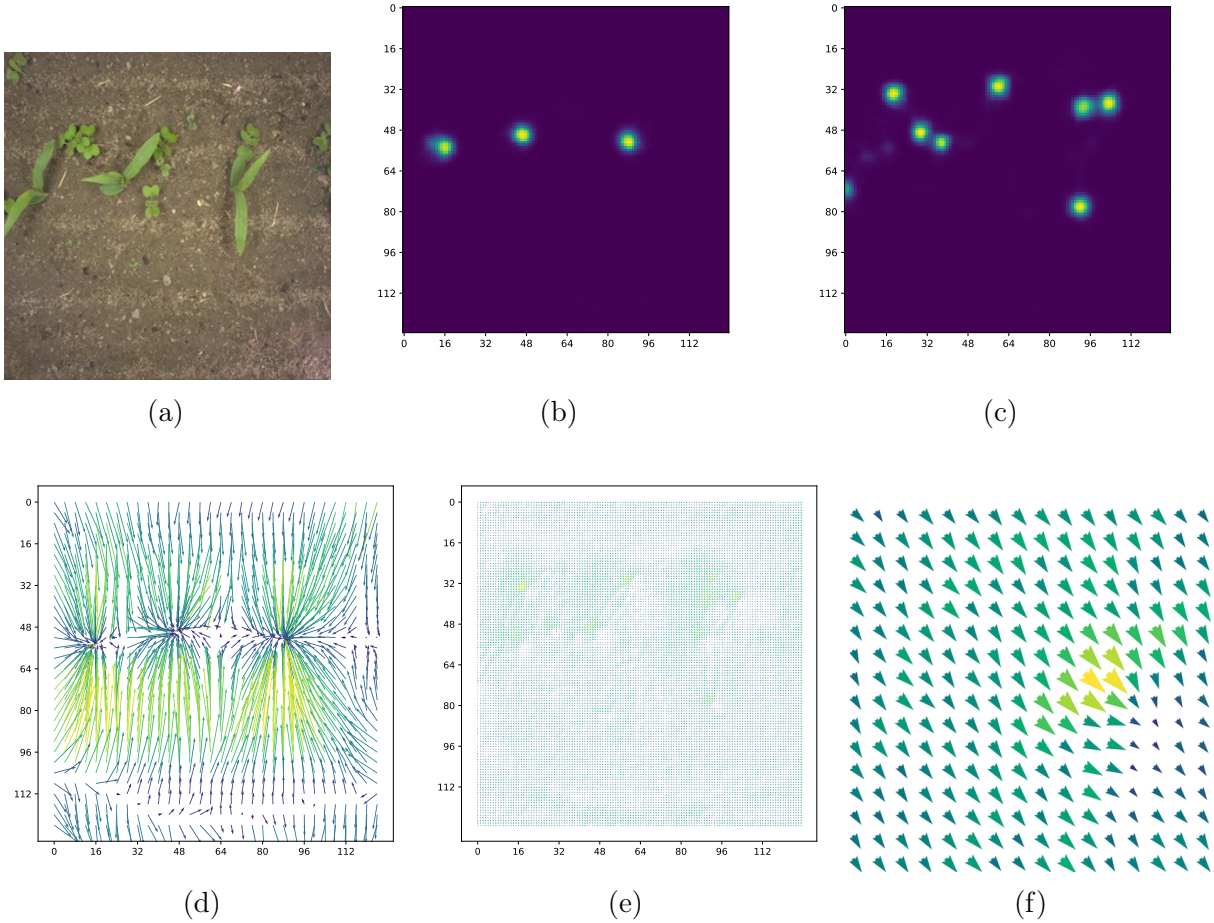


FIGURE 3.9 : Illustration de tenseurs prédits $\hat{Y}$, $\hat{E}$ et $\hat{O}$ par SDNet sur une image de maïs. De gauche à droite et de haut en bas : image d'entrée du réseau (a), carte de chaleur de l'ancre de maïs (b), carte de chaleur de la partie de feuille (c), champ vectoriel d'indices de regroupement (d), champ vectoriel de compensation de la décimation (e) et zoom sur une petite zone du champ de compensation (f).

seuil de distance max_{dist} et le nombre minimal de détections min_{dets} est présentée. Les valeurs optimales déterminées dans cette section seront ensuite utilisées pour comparer les performances des algorithmes dans la section 3.4.

3.3.1 Hyper-paramètres de YOLO

Quatre expérimentations sur les réseaux de la famille YOLO sont présentées dans cette sous-section :

- Une analyse du compromis vitesse-précision de plusieurs variantes de YOLO plus ou moins profondes.
- Une recherche exhaustive du seuil de confiance sur les détections s_c optimal.
- Une recherche exhaustive sur la taille de la boîte englobante de point d'intérêt de tige (voir la section 2.2.1).

Réseau	Profondeur	AP	AP_{50}	AP_{75}	AR_{100}	aIoU	Vitesse
Y4	53 couches	53,9 %	89,7 %	54,6 %	61,2 %	81,0 %	6 IPS
YT4	19 couches	47,3 %	86,3 %	44,8 %	55,4 %	78,1 %	44 IPS
YT3	19 couches	38,8 %	82,3 %	31,6 %	48,6 %	75,4 %	44 IPS

TABLEAU 3.4 : Comparaison des métriques COCO [86] pour trois réseaux YOLO évalués et entraînés sur la base de données DB_1. La vitesse d'inférence, mesurée en images par secondes (IPS), est calculée sur un NVIDIA Jetson Xavier. La définition des images d'entrée est de 544 × 544, le seuil de confiance de 25 % et la taille de boîte de tige de 7,5 %.

- Une recherche exhaustive sur la définition des images en entrée du réseau à l'entraînement.

Ces expérimentations sont réalisées de manière indépendante, e.g. l'impact sur la performance de la taille de boîte englobante est supposé indépendant de l'impact de la définition d'image en entrée. Ceci permet de limiter considérablement le nombre d'entraînements de réseaux[11]. Les autres paramètres et hyperparamètres du réseau et de l'entraînement sont gardés à leur valeur par défaut.

3.3.1.1 Profondeur du réseau

Trois réseaux de profondeurs différentes ont été évalués. La profondeur du réseau impacte non seulement la précision de la détection mais aussi la vitesse d'exécution : un réseau plus profond sera généralement plus performant en termes de précision mais il sera aussi plus lent. Pour plus de détail, voir la section 2.2.2. Le framework Darknet propose principalement deux variantes : YOLOv4 (Y4) qui est le réseau le plus profond et YOLOv4 Tiny (YT4), une variante moins profonde et donc plus rapide. Darknet propose aussi les anciennes versions de YOLO, comme la version 3 par exemple. Cette section propose d'analyser le compromis vitesse-précision entre Y4 et YT4 et d'illustrer le gain de performance introduit par la version 4 de YOLO en comparant YT4 à la variante YOLOv3 Tiny (YT3).

Les trois variantes sont entraînées sur la base de données DB_1 (détection de plantes et de tiges de maïs, de haricot et de poireau) dans les mêmes conditions expérimentales, i.e. les durées d'entraînement et autres paramètres des réseaux sont identiques. En particulier, la définition des images en entrée est de 544 × 544 pixels et la taille de boîte englobante est de 7,5 % du côté le plus petit de l'image (l'impact de ces valeurs est analysé dans la suite). La vitesse d'exécution est mesurée sur un ordinateur de calcul embarqué NVIDIA Jetson Xavier qui reflète une utilisation en condition réelle sur le bloc-outil BIPBIP.

Le tableau 3.4 présente pour les trois variantes de réseau, cinq métriques de détection issues de COCO [86] ainsi que la vitesse d'exécution mesurée en images par seconde (IPS). De manière cohérente, le réseau le plus profond Y4 obtient les meilleures performances de

[11]Le nombre d'entraînements nécessaires est donc proportionnel du nombre de pas de calcul de la recherche exhaustive plutôt que quadratique.

détection (toutes métriques confondues), et le réseau à la fois le moins profond et le plus ancien YT3 obtient les moins bonnes performances. D'une part, ce tableau montre que la nouvelle version YT4 introduit un gain de performance significatif (+8,5 % AP) par rapport à l'ancienne génération YT3 tout en gardant la même vitesse d'inférence (44 IPS). D'autre part, il montre que le gain de performance obtenu par Y4 par rapport à YT4 (+6,6 %) se fait au détriment de la vitesse d'exécution, YOLOv4 étant près de 7 fois plus lent que la version Tiny. Ces résultats illustrent parfaitement les considérations sur le compromis vitesse-précision faites dans la section 2.2.2 : un réseau plus profond est plus précis mais plus lent et un meilleur réseau est caractérisé par une meilleure précision à une vitesse d'exécution au minimum égale.

La vitesse d'exécution de Y4 est trop faible pour une application en temps réel nécessitant au moins 10 IPS (fréquence d'acquisition du module vision de BIPIBP). À l'inverse, les 44 IPS de YT4 sont suffisantes et laissent de la marge de manœuvre pour exécuter d'autres algorithmes en parallèle sur le GPU. Néanmoins, d'autres paramètres tels que la définition d'entrée du réseau permettent de jouer sur le compromis vitesse-précision si besoin (voir la sous-section 3.3.1.4). YT4 est utilisé pour les expérimentations présentées dans la suite.

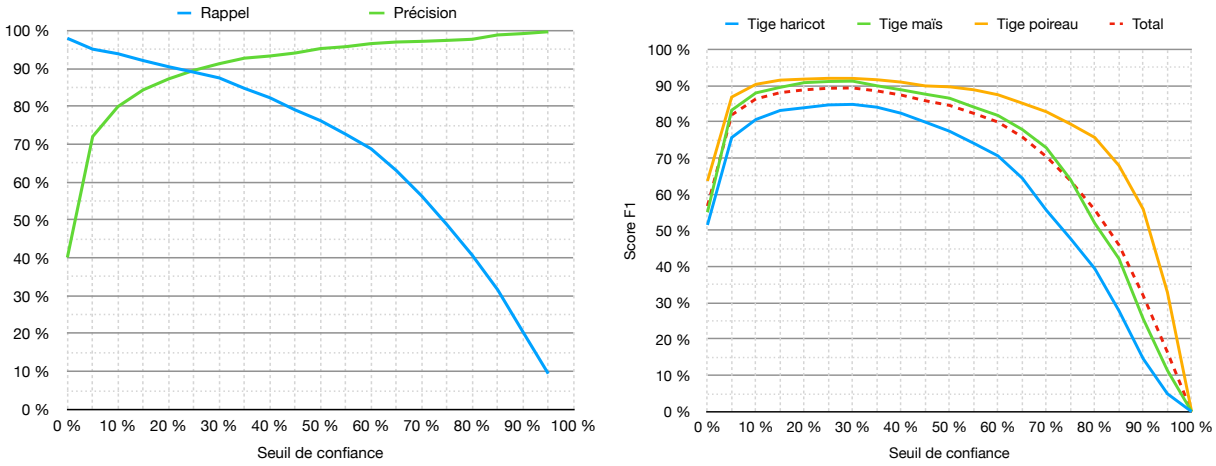
3.3.1.2 Seuil de confiance optimal

Le seuil de confiance s_c contrôle généralement le compromis entre le Rappel et la Précision d'un détecteur. Un seuil bas ne filtre aucune détection, il y aura donc beaucoup de faux positifs et plus de vérités de terrain seront détectées correctement. Au contraire, un seuil élevé filtre la majorité des détections, il y aura peu de faux positifs mais beaucoup d'oublis. Le Rappel diminue et la Précision augmente lorsque le seuil augmente.

Pour vérifier cette affirmation et mesurer l'influence du seuil de confiance s_c sur la performance du détecteur YOLOv4 Tiny, une recherche exhaustive avec un seuil de confiance variant de 0 % à 100 % par pas de 5 % a été menée. Pour cette expérimentation, le réseau YT4 est entraîné sur la base de données DB_1 et les métriques de localisation de points d'intérêt sont utilisées pour quantifier la performance de la détection des tiges (maïs, haricot et poireau). Comme pour l'expérimentation précédente, la définition des images en entrée du réseau est de 544 × 544 et la taille de boîte englobante de tige est de 7,5 % du côté le plus petit de l'image.

La figure 3.10a présente la variation du Rappel et de la Précision (toutes classes confondues) en fonction du seuil de confiance s_c. Le comportement typique du compromis Rappel-Précision est observé : le Rappel diminue de presque 100 % à 10 % tandis que la Précision passe de 40 % à 100 % lorsque le seuil de confiance augmente dans l'intervalle 0 % à 95 % (un seuil de confiance de 100 % donne un Rappel et une Précision nuls[12] car

[12]En prenant pour convention que la Précision est de 0 % s'il y a des vérités de terrain dans la base de validation et 100 % sinon.



(a) Rappel et Précision

(b) Score F1

FIGURE 3.10 : Rappel, Précision et Score F1 en fonction du seuil de confiance s_c du détecteur YOLOv4 Tiny. Le Rappel et la Précision sont présentés toutes classes confondues (moyenne). La définition du réseau est de 544 × 544 et la taille de boîte englobante 7,5 %.

il n'y a aucune détection). Un compromis équitable (Rappel et Précision de 90 %) est obtenu pour une valeur de seuil de confiance de 25 %.

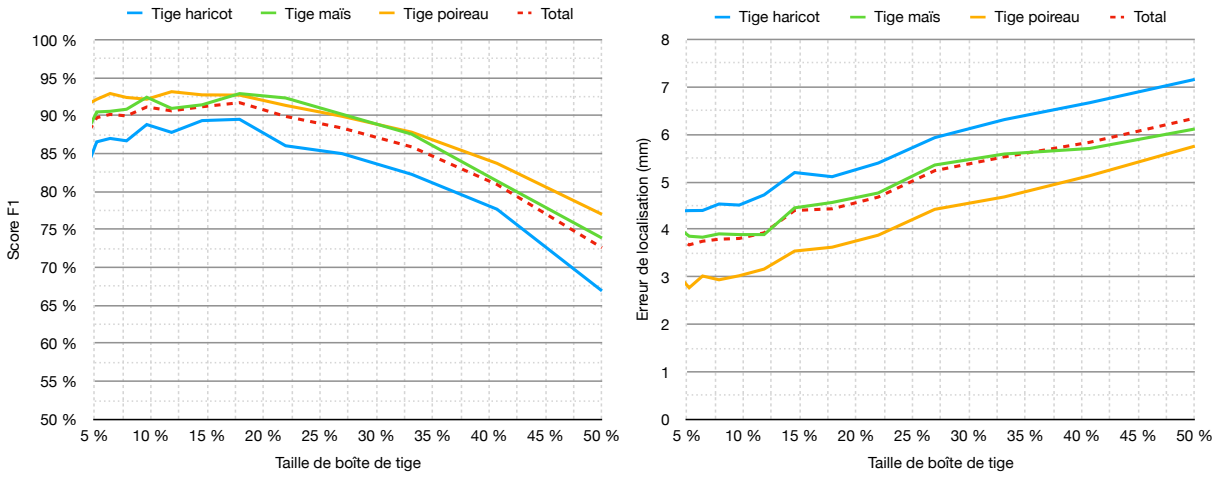
La figure 3.10b présente la variation de la métrique Score F1 détaillée par type de tige en fonction de la valeur du seuil de confiance. Le comportement est similaire pour les trois types de tiges : autour d'un seuil de 0 %, le Score F1 est bas (environ 60 % au total) puis il augmente et atteint un maximum autour d'un seuil de confiance de 25 % (Score F1 de 90 % au total) et enfin il diminue pour atteindre une valeur nulle à un seuil de 100 %. Autour de la valeur optimale de 25 %, le Score F1 varie peu et reste au-dessus de 85 % pour un seuil situé entre 10 % et 50 %. Cependant, le compromis Rappel-Précision varie significativement comme le montre l'analyse précédente. La tige de poireau est mieux détectée que la tige de maïs qui est mieux détectée que la tige de haricot, mais les observations restent valables pour les trois types de tigess (une comparaison entre types de plantes est présentée en détail dans la section 3.4.1).

Un seuil de confiance de 25 % semble optimal puisqu'il offre le meilleur Score F1 et que le Rappel et la Précision sont égaux à cette valeur. Dans la suite des expérimentations, un seuil de 25 % est utilisé pour calculer les métriques.

3.3.1.3 Taille de la boîte englobante de tige

Comme présenté en section 2.2.1, utiliser un détecteur d'objets comme détecteur de points d'intérêt nécessite de définir une taille de boîte englobante arbitraire puisqu'un point d'intérêt n'a pas de taille. Malgré la flexibilité et la robustesse des réseaux de neurones, la taille de la boîte englobante de tige de plante sélectionnée peut potentiellement affecter l'entraînement du réseau et impacter les performances.

Pour mesurer l'influence de ce paramètre, vingt réseaux YOLOv4 Tiny ont été entraînés



(a) Score F1

(b) MAE_{loc}

FIGURE 3.11 : Score F1 et MAE_{loc} (mm) en fonction de la taille de boîte de tige de YOLOv4 Tiny comparé par type de tige (échelle logarithmique).

à des valeurs de taille de boîte englobante variant de 1 % à 50 % sur une échelle logarithmique. La taille de la boîte englobante est exprimée en pourcentage du côté de l'image le plus petit[13] (la hauteur dans notre cas puisque les images sont toutes approximativement au format 4 :3). Comme pour les expérimentations précédentes, les réseaux sont entraînés sur la base de données DB_1 et les conditions expérimentales sont les mêmes à l'exception de la taille de la boîte de tige : la définition des images en entrée est de 544 × 544 et le seuil de confiance est de 25 %. Les métriques de localisation de points d'intérêt sont utilisées.

La figure 3.11 présente la variation du Score F1 en fonction de la taille de la boîte de tige ventilé par type de plante. En dessous de 5 % (non représenté) le Score F1 total chute à des valeurs proches de 0 %. Entre 5 % et 25 % le Score F1 total reste stable autour de 90 % puis il diminue doucement au-delà de 25 % pour atteindre une valeur d'environ 70 % pour une taille de boîte de tige de 50 %. L'évolution du Score F1 est similaire pour les trois types de tiges et montre que le haricot est moins bien détecté que le maïs et le poireau.

La figure 3.11b présentant la variation de la métrique MAE_{loc} en fonction de la taille de boîte de tige dresse un constat similaire. En dessous de 5 % le MAE_{loc} n'est pas défini car le réseau ne converge pas et ne détecte rien. Entre 5 % et 50 % le MAE_{loc} augmente de manière linéaire et passe de 4 mm à 6 mm en moyenne. Là aussi le comportement est similaire pour les trois types de tiges et montre que le haricot est moins bien localisé que les autres tiges.

La taille de la boîte a donc une influence sur la performance de la localisation de tiges mais celle-ci reste faible. Une valeur très faible de taille de boîte englobante (inférieure à 5 %) ne permet pas une convergence. Une valeur très élevée (supérieure à 50 %) dégrade

[13]Ceci permet de normaliser la taille de la boîte englobante et de ne pas dépendre de la définition de l'image.

significativement les performances. Cependant, entre 5 % et 25 % la performance reste remarquablement stable malgré une légère baisse aux valeurs élevées. Ces résultats illustrent la flexibilité et la robustesse des réseaux de neurones profonds : non seulement un réseau de détection d'objets peut être utilisé comme détecteur de points d'intérêt de manière fiable, mais en plus, il se révèle peu sensible à la variation de la taille de la boîte englobante arbitraire sélectionnée pour modéliser les points d'intérêt.

En pratique, une taille de boîte de 7,5 % donne un Score F1 élevé (90 % toutes plantes confondues) et une excellente $\mathrm{MAE}_{\mathrm{loc}}$ (un peu moins de 4 mm) donc cette valeur est utilisée dans la suite des expérimentations. Cependant, d'autres valeurs de taille entre 5 % et 25 % conduisent aussi à de bons résultats malgré un compromis différent entre le Score F1 et le $\mathrm{MAE}_{\mathrm{loc}}$ (par exemple une taille de boîte de 17 % donne un Score F1 supérieur de 1 % au détriment d'une baisse de $\mathrm{MAE}_{\mathrm{loc}}$ de 1 mm).

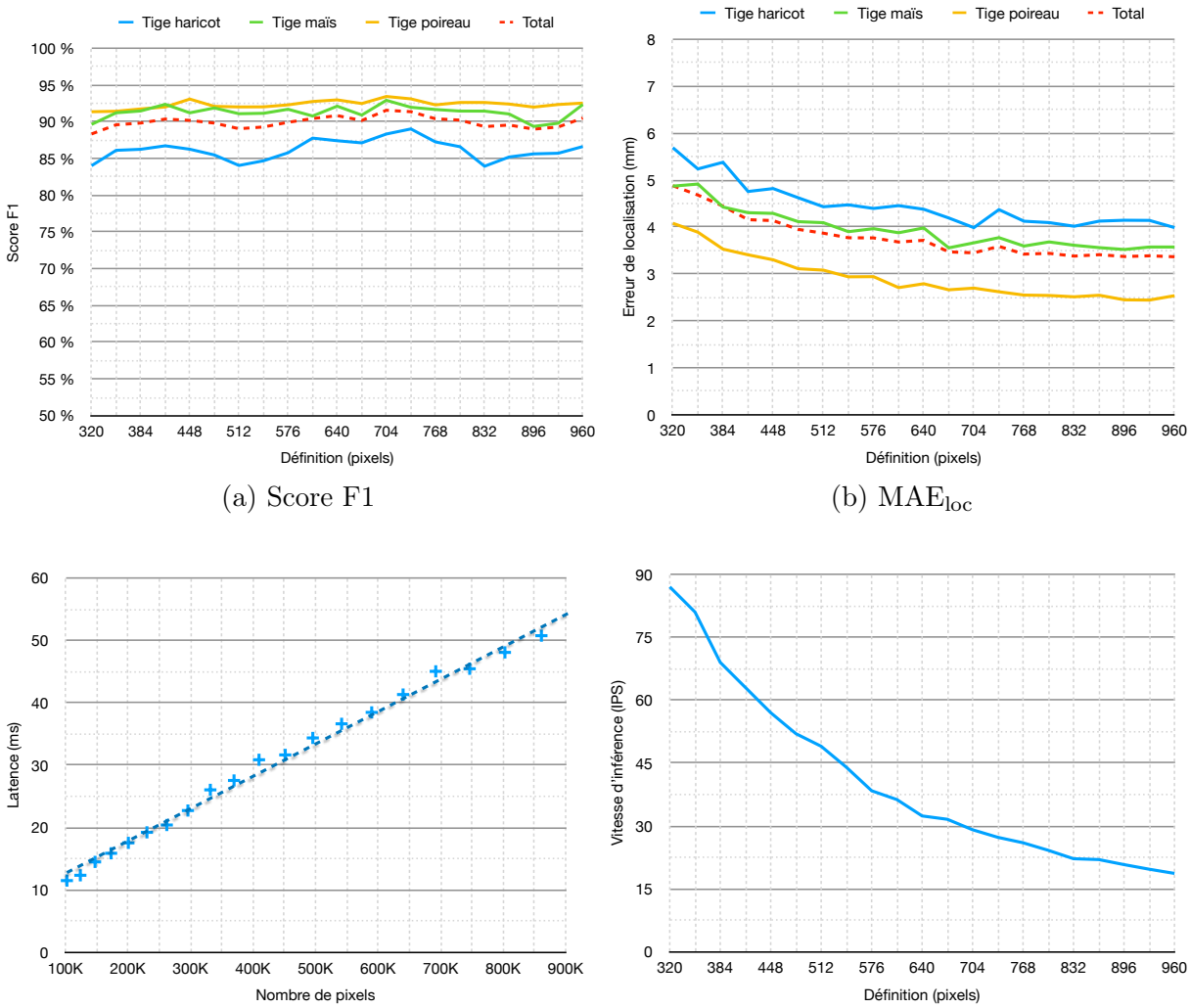
3.3.1.4 Définition d'entrée du réseau

Un réseau de neurones est usuellement entraîné à une définition d'image fixe[14], le plus souvent avec rapport hauteur/largeur (*Aspect Ratio*) de 1 pour 1 (images carrées). La définition a une influence directe sur le nombre d'opérations requis pour l'exécution du réseau (entraînement et inférence) et donc sur la vitesse d'inférence qu'il est possible d'atteindre. De plus, une définition plus haute permet d'avoir une image plus détaillée pouvant potentiellement améliorer les performances de détection du réseau. Comme pour la profondeur du réseau, la définition contrôle donc un compromis entre la vitesse d'exécution et la performance de la détection et de la localisation.

Pour mesurer son impact sur la performance, une recherche exhaustive sur la définition d'entrée du réseau YT4 a été menée. Vingt-et-un réseaux ont été entraînés sur la base de données DB_1 avec une définition d'entrée allant de 320×320 pixels à 920×920 pixels et sont évalués avec les métriques de localisation de points d'intérêt. Les autres paramètres sont constants : la taille de boîte de tige est de 7,5 % et le seuil de confiance est de 25 %.

La figure 3.12a présente le Score F1 en fonction de la définition des images d'entrée de TY4 ventilé par type de tige. Les courbes de Score F1 présentent un profil plat quel que soit le type de plante, malgré un comportement plus chaotique pour le haricot. Le Score F1 total évolue sur une plage de variation de 2 % maximum centrée sur 90 %. Il ne semble pas y avoir de valeur optimale de la définition d'entrée. La figure 3.12b présente le $\mathrm{MAE}_{\mathrm{loc}}$ en fonction de la définition des images d'entrée du réseau étudié. Le $\mathrm{MAE}_{\mathrm{loc}}$ total diminue légèrement sur la plage de variation et passe de 5 mm pour une définition de 320 pixels de côté à environ 3,5 mm pour une définition de 960 de côté. Une définition plus haute améliore donc légèrement la localisation des points d'intérêt. Comme précédemment, le comportement est similaire pour les trois types de tiges mais montre que le haricot est

[14]Hormis certaines augmentations de données qui peuvent faire fluctuer la définition des images autour de cette valeur fixe pour émuler un changement d'échelle.



(a) Score F1

(b) $\mathrm{MAE}_{\mathrm{loc}}$

(c) Latence

(d) Vitesse d'inférence

FIGURE 3.12 : Score F1, $\mathrm{MAE}_{\mathrm{loc}}$ (mm) et vitesse d'inférence en fonction de la définition d'entrée de YT4. La vitesse d'inférence est calculée sur un NVIDIA Jetson Xavier.

moins bien localisé que les autres cultures.

La figure 3.12c présente la latence d'inférence calculée sur un NVIDIA Jetson Xavier en fonction du nombre de pixels (hauteur $\times$ largeur). Il y a une corrélation linéaire positive forte entre ces deux paramètres ($R^2 > 99$ %) qui est attendue puisque la vitesse d'inférence est proportionnelle au nombre d'opérations (*Floating Point Operations* (FLOP)), qui est elle-même proportionnelle au nombre de pixels à traiter[15]. La figure 3.12d présente la vitesse d'inférence (en images par seconde) en fonction de la définition d'entrée. Le réseau avec la plus faible définition (320 pixels de côté) a une vitesse d'inférence d'environ 10 ms (100 IPS) tandis que le réseau avec la définition la plus élevée (960 pixels de côte) a une vitesse d'inférence d'environ 50 ms (20 IPS), soit cinq fois plus lent.

L'impact de la définition est donc relativement faible sur les performances de détection de YT4. Il n'y a pas de gain significatif en détection (Score F1) et un gain relativement

[15]Le bloc de base des réseaux de neurones convolutifs (CNN) est la convolution qui est une opération donc la complexité calculatoire est en $\mathcal{O}(W \times H)$ où W et H sont les dimensions de l'image.

Paramètre	Plage de variation	Valeur retenue	Commentaire
Profon-deur/version	YT3, YT4, Y4	YT4	Compromis vitesse-précision, impact fort
Seuil de confiance	0 % à 100 %	25 %	Compromis Rappel-Précision, impact modéré
Taille de boîte	1 % à 50 %	7,5 %	Faible impact, peu de variation
Définition d'entrée	320 à 960 pixels de côté	544	Compromis vitesse-précision, faible impact

TABLEAU 3.5 : Synthèse de l'impact des paramètres principaux de YOLO sur la performance.

faible en localisation (1,5 mm). La définition d'entrée des images ne semble pas être un critère déterminant puisqu'elle n'améliore que marginalement les performances. En revanche, la définition a un impact prédominant sur la vitesse d'inférence du réseau qui est proportionnelle au nombre de pixels à traiter.

Le choix d'une valeur de définition est donc principalement guidé par des considérations de vitesse d'exécution puisque le gain apporté sur la performance de la détection par une plus haute définition est marginal. Toutes les définitions testées conviennent en termes de vitesse d'inférence puisqu'elles sont supérieures aux 10 IPS demandées par le bloc-outil BIPBIP. En pratique, une définition de 544 $\times$ 544 est choisie, elle permet d'atteindre une vitesse d'exécution de 44 IPS largement suffisante tout en accordant un petit gain de localisation de 1 mm (au-delà, le gain ne serait que de 0,5 mm).

3.3.1.5 Synthèse

Les expérimentations sur les réseaux de la famille YOLO présentées dans cette sous-section ont permis d'analyser l'influence des principaux paramètres sur les performances : la profondeur du réseau, le seuil de confiance, la taille de la boîte englobante de tige et définition des images d'entrée. Le tableau 3.5 synthétise les paramètres évalués et les valeurs optimales sélectionnées.

En résumé, utiliser un réseau de détection d'objets pour de la détection de points d'intérêt se révèle viable et permet d'obtenir une bonne performance et une vitesse d'exécution suffisante pour une application en temps réel. Le réseau est remarquablement robuste à la taille de la boîte de tige, qui a un faible impact sur la performance sur une large plage de valeur. La profondeur du réseau et la définition des images d'entrées contrôlent essentiellement un compromis entre la précision de la détection et la vitesse d'inférence : leurs valeurs doivent donc être choisies en fonction de l'application visée et du matériel de calcul. Le seuil de confiance contrôle quant à lui un compromis entre le Rappel et la Précision. Une valeur optimale existe malgré une plage de variation assez large.

3.3.2 Paramètres de SDNet

Trois expérimentations sur le réseau de neurones profond SDNet sont présentées dans cette sous-section :

- Une recherche exhaustive conjointe sur le seuil de confiance t_c et le seuil de distance t_d du décodeur.
- Une recherche exhaustive sur la taille du noyau gaussien de l'encodeur servant à créer les cartes de chaleur.
- Une analyse de l'amélioration des performances apportée par les vecteurs de compensation de la décimation.

Les considérations sur la profondeur et la définition faites dans la sous-section précédente ont montré que ces paramètres contrôlent principalement le compromis entre la vitesse d'inférence et la performance de détection. Ces observations ne sont pas propres aux réseaux de détection d'objets et SDNet est conçu sur une architecture neuronale semblable : il est donc pertinent d'extrapoler les conclusions précédentes pour SDNet. Afin d'assurer une comparaison équitable avec YOLOv4 Tiny, une profondeur et une définition similaire sont choisies. Ainsi, la colonne peu profonde "resnet34" [56] est utilisée et la définition d'entrée est fixée à 512×512[16].

3.3.2.1 Seuil de confiance et de distance du décodeur

Le seuil de confiance pilote le compromis Rappel-Précision du réseau (voir section 3.3.1.2) : une valeur basse est associée à un Rappel élevé et à une Précision basse, et vice-versa. Le seuil de distance t_d du décodeur contrôle quant à lui l'association des points d'intérêt. Un seuil faible rejettera la majorité des associations et génèrera des plantes avec des feuilles manquantes. Au contraire, un seuil élevé pourrait générer des plantes avec trop de feuilles ou avec des feuilles d'autres plantes.

Pour analyser l'impact du seuil de confiance t_c et du seuil de distance t_d, une recherche exhaustive conjointe[17] sur ces deux paramètres a été menée. Le seuil de confiance varie de 0 % à 100 % par pas de 5 % et le seuil de distance varie de 0 % à 50 % par pas de 2,5 %. L'entraînement est réalisé sur la base de données DB_2 de structure de plante et les métriques d'estimation du nombre de feuilles sont utilisées pour évaluer les performances[18]. La taille du noyau gaussien de l'encodeur des cartes de chaleur est fixée à 5 % (plus de détail dans la sous-section suivante).

[16]Cette définition est choisie uniquement pour les recherches exhaustives présentées dans cette sous-section. Une comparaison équitable avec YOLOv4 Tiny (qui utilise la colonne peu profonde "Darknet-19" [112]) utilisant la même définition d'image d'entrée que SDNet est présentée dans la section 3.4.1.

[17]Cette recherche ne nécessite pas d'entraînement de réseau et est donc peu coûteuse en temps de calcul.

[18]Le seuil de distance t_d contrôle l'association des feuilles aux tiges, il n'a dont pas d'impact sur la capacité de localisation des points d'intérêt, d'où l'utilisation de la métrique d'estimation du nombre de feuilles.

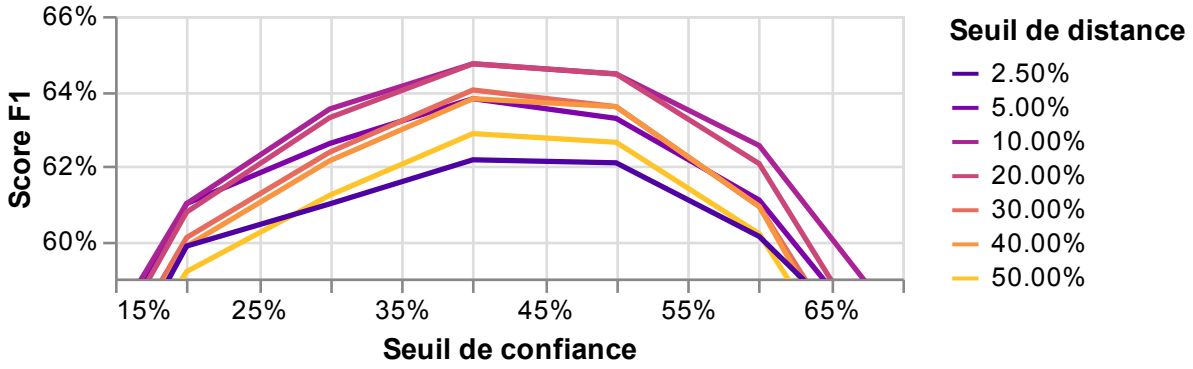


FIGURE 3.13 : Recherche exhaustive conjointe sur les paramètres t_c et t_d du décodeur de SDNet.

La figure 3.13 présente l'évolution du Score F1 en fonction du seuil de confiance et du seuil de distance. L'évolution du Score F1 en fonction du seuil de confiance est sensiblement le même suivant les valeurs du seuil de distance (couleur des courbes). Les courbes décrivent une trajectoire en cloche similaire à la trajectoire de celle de YOLO décrite dans la section 3.3.1.2 et par extrapolation les causes sont probablement les mêmes : un seuil bas ne filtre pas les points d'intérêt de mauvaise qualité, ce qui génère beaucoup de faux positifs (la Précision est basse), et un seuil haut filtre la majorité des points d'intérêt générant beaucoup d'oublis (Rappel bas). En dessous d'un seuil de confiance de 20 % et au-dessus de 70 % (non représenté pour améliorer l'intelligibilité de la figure), le Score F1 chute rapidement vers 0 %. Le maximum de Score F1 (entre 62 % et 65 % selon les courbes) est atteint pour un seuil de confiance de 40 %. Cependant, des valeurs de seuil de confiance entre 30 % et 50 % conduisent à des performances sensiblement proches de la valeur optimale.

L'évolution du Score F1 en fonction du seuil de distance suit aussi une trajectoire de courbe en cloche. Une valeur inférieure à 2,5 % (non représentée) détériore fortement les performances qui sont alors proches de 0 %. Le Score F1 augmente progressivement entre un seuil de distance de 2,5 % et 10 % où un optimum est atteint (courbe la plus haute enveloppant les autres). Le score redescend ensuite doucement vers un Score F1 maximal de 63 % pour un seuil de distance de 40 %. Au-delà de 50 % (non représenté) le Score F1 sature et se stabilise à l'endroit de la courbe jaune (Score F1 maximal de 63 %).

Cette analyse montre que le seuil de confiance a un impact sur la performance de SDNet. Une valeur de 40 % conduit à une valeur optimale de Score F1 quel que soit le seuil de distance. Cependant, la variation du Score F1 est minimale sur une plage étendue, il ne varie que d'environ 2 % sur une plage allant de 20 % à 60 %. La performance ne se dégrade réellement qu'en dessous de 20 % et au-dessus de 60 %, quel que soit le seuil de distance.

En revanche, le seuil de distance a un impact plus modéré sur la performance. La variation de Score F1 est au maximum de 3 % (atteinte au seuil de confiance de 40 %)

pour des valeurs de seuil de distance entre 2,5 % et 50 % (et même 100 %). Seules des valeurs inférieures à 2.5 % dégradent significativement la performance. Pour expliquer la saturation présente à des seuils élevés, il faut noter qu'un seuil de 100 % est équivalent à une absence de filtrage sur l'association des feuilles aux tiges : les points d'intérêt de feuille sont simplement associés au point d'intérêt de tige le plus proche. Ce mode de fonctionnement sans filtrage n'est pas aberrant, c'est d'ailleurs l'approche employée par [142] pour l'estimation de la pose humaine. Cette expérimentation montre donc qu'un filtrage supplémentaire sur la qualité de l'association permet d'augmenter les performances de détection (environ 2 % de Score F1).

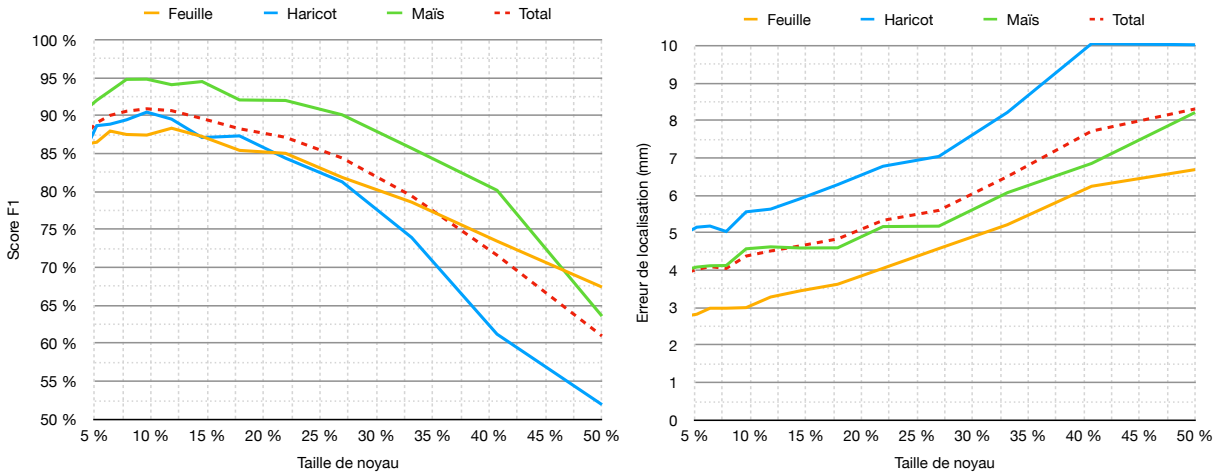
Cette recherche exhaustive conjointe sur les seuils du décodeur de SDNet montre qu'un seuil de confiance de 40 % et un seuil de distance de 10 % permettent d'obtenir les meilleures performances. Ces valeurs sont utilisées pour les expérimentations présentées dans la suite.

3.3.2.2 Taille du noyau gaussien de l'encodeur

La taille du noyau gaussien de l'encodeur de SDNet permet de moduler l'extension spatiale des courbes gaussiennes qui représentent les points d'intérêt dans les cartes de chaleur. Ce paramètre est analogue à la taille de boîte englobante de tige qui nécessite d'être fixée pour YOLO. Cependant, ce paramètre n'est pas "arbitraire" contrairement à YOLO et son impact sur l'entraînement est plus facilement explicable. En effet, il permet de définir une incertitude sur la position d'un point d'intérêt qui pénalisera plus ou moins le réseau (via la fonction de coût) si celui-ci prédit le point d'intérêt à une position inexacte.

Afin de mesurer l'influence de ce paramètre, vingt réseaux SDNet ont été entraînés sur la base de données DB_2 avec une valeur de taille de noyau de l'encodeur différente. La taille d'une courbe gaussienne représentant un point d'intérêt est définie comme la largeur de la courbe à trois σ_e (écart-type du noyau gaussien de l'encodeur). Comme pour YOLO, cette taille est exprimée en pourcentage du côté le plus petit de l'image afin de normaliser la taille en fonction de la définition des images d'entrée. Les valeurs de taille testées sont étalées entre 1 % et 50 % selon une échelle logarithmique. Les autres paramètres de SDNet sont gardés constants : le seuil de confiance est de 40 % et le seuil de distance est de 10 %. Les métriques de localisation de points d'intérêt sont utilisées pour quantifier les performances puisque la taille du noyau gaussien de l'encodeur a une influence directe sur l'apprentissage des points d'intérêt.

La figure 3.14a présente la variation du Score F1 en fonction de la taille du noyau gaussien de l'encodeur de SDNet. Le comportement par type de point d'intérêt est sensiblement le même : les courbes décrivent une courbe en cloche. En dessous d'une taille de 5 % (non représenté) la convergence du réseau n'est pas assurée et le Score F1 est faible ou nul. Ensuite, un optimum est atteint pour une taille de noyau de 10 % qui conduit à un Score F1 total (toutes plantes confondues) de 90 %. Finalement, à partir d'une taille



(a) Score F1

(b) $\mathrm{MAE}_{\mathrm{loc}}$

FIGURE 3.14 : Score F1 et $\mathrm{MAE}_{\mathrm{loc}}$ (mm) en fonction de la taille du noyau Gaussien de SDNet comparé par type de point d'intérêt.

de 10 % et jusqu'à 50 % le Score F1 décroit progressivement pour atteindre une valeur de 60 %.

La figure 3.14b présente la variation du score de localisation $\mathrm{MAE}_{\mathrm{loc}}$ en fonction de la taille du noyau. Une taille de noyau faible donne les meilleures performances en localisation (tous types de points d'intérêt confondus) : la valeur la plus basse de $\mathrm{MAE}_{\mathrm{loc}}$ (4 mm) est obtenue pour la plus petite taille de noyau (5 %). La performance se dégrade ensuite lorsque la taille de noyau augmente et atteint plus de 8 mm pour une taille de 50 %. La valeur optimale est donc atteinte pour la valeur de taille la plus faible possible.

Le même comportement que pour YOLO est donc observé pour SDNet : le Score F1 chute lorsque la taille de noyau est très basse (inférieure à 5 %) et diminue progressivement quand la taille est élevée (au-dessus de 15 %). Entre 5 % et 15 % la performance est relativement stable mais contrairement à YOLO, la plage de stabilité est plus restreinte et une valeur optimale se distingue plus clairement. Le comportement de la performance en localisation est aussi similaire : elle diminue de manière monotone lorsque la taille de noyau augmente. Le compromis idéal semble être situé autour d'une taille de 10 % puisque le Score F1 est maximal à cette valeur (un peu plus de 90 %) et que le $\mathrm{MAE}_{\mathrm{loc}}$ est proche de la valeur minimale (4,5 mm). Cette valeur de taille est utilisée dans la suite.

3.3.2.3 Vecteur de compensation de la décimation

La compensation de la décimation est une technique utilisée dans plusieurs articles présentant des détecteurs de pose basés sur la détection de points d'intérêt via des cartes de chaleur comme [142]. Cependant, le gain de performance supposé est rarement analysé ou commenté dans la littérature. Cette expérimentation propose de quantifier le gain de performance apporté par les vecteurs de compensation de SDNet en évaluant le même réseau entraîné avec et sans les vecteurs de compensation. Dans la version sans, les vecteurs

Compensation	Label	Rappel	Précision	Score F1	MAE_{loc}
Avec	Haricot	85,7 %	92,3 %	88,8 %	5,1 ± 0,28 mm
	Maïs	93,1 %	96,0 %	94,5 %	3,7 ± 0,19 mm
	Feuille	86,1 %	91,8 %	88,8 %	2,8 ± 0,08 mm
	Total	87,0 %	92,4 %	89,6 %	3,3 ± 0,08 mm
Sans	Haricot	85,7 %	92,3 %	88,8 %	6,0 ± 0,27 mm
	Maïs	93,1 %	96,0 %	94,5 %	4,6 ± 0,20 mm
	Feuille	86,1 %	91,7 %	88,8 %	4,3 ± 0,09 mm
	Total	87,1 %	92,4 %	89,6 %	4,3 ± 0,08 mm

TABLEAU 3.6 : Gain de performance en localisation de points d'intérêt apporté par la compensation de décimation de SDNet comparé par type de point d'intérêt.

de compensation de décimation sont simplement ignorés pendant le processus de décodage. Ces vecteurs de compensation affectent principalement la précision en localisation des points d'intérêt : la métrique de localisation est donc employée pour mesurer l'impact sur la performance. Les autres paramètres de SDNet sont gardés constants : le seuil de confiance est de 40 %, le seuil de distance est de 10 % et la taille de noyau est de 10 %.

Le tableau 3.6 présente les métriques de localisation de points d'intérêt par type de point d'intérêt avec et sans la compensation de décimation. Elle montre des résultats presque identiques quel que soit le type de point d'intérêt, pour le Rappel, la Précision et le Score F1. Seule la métrique de localisation MAE_{loc} évolue, la compensation de la décimation apporte un gain en localisation de 1 mm pour tous les types de points d'intérêt (soit environ 1,3 pixels de l'image d'entrée de taille 512 × 512). La compensation de la décimation améliore donc légèrement les performances en localisation mais n'augmente pas la performance de la détection des points d'intérêt.

Le gain apporté par la compensation de la décimation est modéré mais il n'impose pas de pénalité sur le temps d'exécution de SDNet, son coût calculatoire étant négligeable devant le coût calculatoire des couches convolutives du réseau. La compensation de la décimation est donc conservée.

3.3.2.4 Synthèse

Le tableau 3.7 synthétise les résultats produits par le biais des recherches exhaustives sur les paramètres principaux de SDNet. Comme pour YOLO, le seuil de confiance contrôle un compromis entre le Rappel et la Précision. Sa valeur optimale est de 40 %. Le seuil de distance a un impact positif sur la performance d'estimation du nombre de feuilles (comparé à une absence de seuil), mais celui-ci reste modéré et la performance varie peu selon la valeur du seuil. La valeur optimale est de 10 %. L'influence de la taille du noyau est similaire à l'influence de la taille de la boîte de tige pour YOLO mais est plus marquée et varie plus fortement. Sa valeur optimale est de 10 %. La compensation de décimation apporte un léger gain de performance de localisation sans ralentir l'exécution du réseau.

Paramètre	Plage de variation	Valeur retenue	Commentaire
Seuil de confiance t_c	0 % à 100 %	40 %	Compromis Rappel-Précision, impact modéré
Seuil de distance t_d	0 % à 50 %	10 %	Impact faible
Taille du noyau	1 % à 50 %	10 %	Impact modéré
Compensation de la décimation	avec et sans	avec	Impact faible, pas d'impact sur la vitesse d'exécution

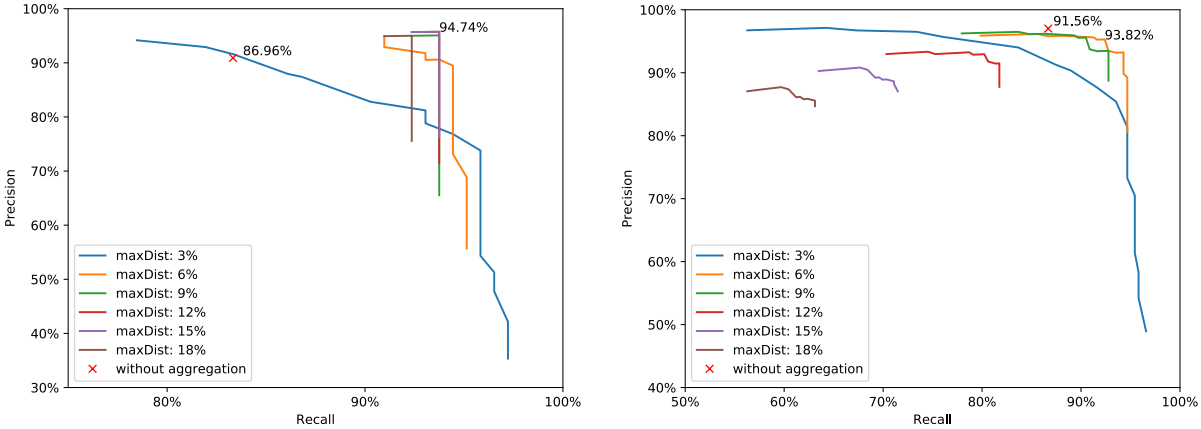
TABLEAU 3.7 : Synthèse de l'impact des paramètres principaux de SDNet sur la performance.

3.3.3 Paramètre de l'agrégation temporelle

L'algorithme d'agrégation temporelle des points d'intérêt de tige est tributaire de deux paramètres principaux : le seuil de distance max_{dist} pour qu'une détection soit associée à un tracker, et le seuil min_{dets} du nombre de détections qu'un tracker doit contenir pour qu'il soit considéré comme valide. Les autres paramètres ont été fixés au préalable de manière heuristique :

- Le seuil sur l'EGI t_e est fixé à 35 pour que le masque de la non-végétation contienne qualitativement peu de fausses détections et peu d'oublis de détection.
- L'opérateur morphologique `morph` consiste en deux dilatations avec un élément structurant en forme de disque d'un diamètre de 10 pixels, un remplissage des trous ayant une aire inférieure à un millième de l'aire de l'image et une suppression des petits objets dont l'aire est inférieure à un millième de l'aire de l'image.
- Les nombreux paramètres de l'estimateur du flot optique dense `flot_optique` ont été réglés pour permettre un calcul stable et rapide (taille et facteur d'échelle de la pyramide d'images, taille des blocs, nombre d'itérations, etc.).
- La durée d'inactivité des trackers $max_{inactive}$ est fixée à 30. Cette durée correspond au nombre moyen d'observations d'une même tige de plante au cours d'une séquence (régie par la vitesse d'avancement du module et la fréquence d'acquisition de la caméra). Cette valeur garantit donc qu'une observation d'une tige réalisée lors de l'entrée dans le champ de vision de la caméra puisse être associée à l'observation de cette même tige lorsqu'elle est sur le point de quitter le champ de vision et qu'aucune autre détection n'a été réalisée entre les deux.

Cette sous-section propose une recherche exhaustive sur les paramètres max_{dist} et min_{dets} de l'algorithme d'agrégation. Le paramètre max_{dist} varie de 3 % à 18 % par pas de 3 % et min_{dets} varie de 1 à 20 par pas de 1. La base de données de séquences d'images DB_3 est utilisée pour évaluer les performances et la métrique de localisation de points d'intérêt



(a) Maïs (b) Haricot

FIGURE 3.15 : Courbes rappel-précision de la recherche exhaustive conjointe sur max_{dist} et min_{dets} de l'algorithme d'agrégation temporelle ventilées par type de tige. Le paramètre min_{dist} est la variable de la courbe et max_{dist} la couleur. Le score F1 de l'algorithme sans agrégation est illustré pour référence par la croix rouge. La valeur de Score F1 optimale pour le maïs (94,74 %) est atteinte pour un seuil de distance de 12 % et un seuil sur le nombre de détections de 10. Pour le haricot, la valeur optimale de Score F1 (93,82 %) est atteinte à un seuil de distance de 6 % et un nombre de détections de 13.

est adoptée puisque l'algorithme d'agrégation est conçu pour améliorer la détection des tiges de plante. Pour générer les détections de tiges nécessaires à l'algorithme d'agrégation, le détecteur YT4[19] est utilisé dans la configuration Y_{hp} avec les paramètres optimaux établis précédemment.

La figure 3.15 présente la variation du Rappel et de la Précision en fonction de min_{dets} (tracé des courbes) et max_{dist} (couleur des courbes) ventilée par type de tige (maïs ou haricot). La performance de YT4 sans l'algorithme d'agrégation est annotée en supplément par une croix rouge. Il apparaît que le paramètre min_{dets} contrôle un compromis entre le Rappel et la Précision quel que soit le type de plante : une valeur faible (1 par exemple, qui correspond aux extrémités droites des courbes) génère beaucoup de faux positifs car elle autorise tous les trackers ayant au moins une détection (c'est-à-dire tous) : le rappel est donc haut mais la précision basse. Au contraire, une valeur élevée (20 par exemple qui correspond à l'extrémité gauche de la courbe) filtre la majorité des trackers et ne laisse que ceux ayant un grand nombre d'observations, le rappel est donc bas et la précision haute.

Le seuil de distance max_{dist} a quant à lui un impact direct sur la performance globale : une variation du seuil de distance entraîne soit une baisse, soit une hausse à la fois du Rappel et de la Précision, et donc du Score F1. La figure 3.15 qu'une valeur de seuil de distance faible (courbe bleue à 3 % et en dessous) conduit à un Score F1 faible et qu'une valeur de seuil de distance élevée (courbe marron à 18 %) donne aussi un Score F1 bas (surtout visible pour le haricot où la courbe marron est en dessous de toutes les autres

[19]Une comparaison entre YT4 et SDNet comme générateurs de détections pour l'agrégation est détaillée dans la suite.

Seuil	Plage de variation	Maïs	Haricot	Commentaire
Distance	3 % à 18 %	12 %	6 %	Dépend du schéma de semis
Nombre de détections	1 à 20	10	13	Dépend de la vitesse d'avancement et de la fréquence d'acquisition

TABLEAU 3.8 : Synthèse de l'impact des paramètres principaux de l'agrégation sur les performances. Le réseau YT4 dans la configuration Y_{hp} est utilisé pour générer les prédictions et les hyperparamètres sont fixés aux valeurs optimales déduites précédemment.

courbes). Une valeur optimale est atteinte entre 3 % et 18 %. Notamment, la courbe orange relative à un seuil max_{dist} de 6 % est au-dessus de toutes les autres et constitue l'optimum.

Cependant, le couple de seuil de distance et de nombre de détections optimal dépend du type de culture. Le Score F1 le plus élevé pour le maïs (94,74 %) est atteint pour un seuil de distance de 12 % et un seuil sur le nombre de détections de 10. Pour le haricot, la valeur optimale de Score F1 (93,82 %) est atteinte à un seuil de distance de 6 % et un nombre de détections de 13. Le seuil de détections est donc assez proche entre les deux types de culture mais le seuil de distance est très différent, celui du haricot (6 %) est deux fois plus petit que celui du maïs (12 %). Cette différence de seuil de distance est à mettre en relation avec la distance du semis d_{within} entre les plantes : elle est plus élevée pour le maïs (environ 15 cm) que pour le haricot (entre 3 cm et 8 cm). L'espacement plus serré des tiges de haricot cause potentiellement plus de confusions entre plantes voisines, un seuil de distance plus faible permet donc d'éviter d'agréger des observations d'autres plantes qui seraient trop proches. Au contraire, les plants de maïs sont plus éloignés les uns des autres donc un seuil de distance plus élevé risque moins de provoquer une agrégation erronée de la tige d'une autre plante.

Cette analyse montre aussi les gains de performance apportés par l'algorithme d'agrégation : une augmentation du Score F1 de 7,78 % pour le maïs et de 2,26 % pour le haricot. Ces résultats sont commentés plus loin dans la section 3.4.2 et ils sont comparés selon l'utilisation du détecteur YT4 ou SDNet pour générer les détections de tige. Les seuils de distance et de nombre de détections optimaux sélectionnés ici sont utilisés dans la suite. Ils sont résumés dans le tableau 3.8.

3.4 Évaluation comparative des méthodes

La section précédente a analysé l'influence des paramètres principaux des méthodes développées et a permis de fixer les valeurs optimales. Cette section propose d'évaluer les algorithmes et de comparer les performances. Plusieurs analyses sont ainsi présentées :

- Dans la section 3.4.1, une analyse et une comparaison des performances de YT4 et SDNet sur les tâches de localisation d'organes et d'estimation du nombre de

feuilles. Les deux réseaux sont entraînés sur la même base de données et dans une configuration similaire afin de les comparer équitablement.

- Dans la section 3.4.2, une analyse de l'amélioration des performances par l'algorithme d'agrégation et une comparaison des performances selon l'utilisation de YT4 ou SDNet pour fournir les détections de tiges.

Pour finir, la section 3.5 discute les résultats et les limites des méthodes explorées.

3.4.1 Comparaison des performances de SDNet et YOLOv4 Tiny

Les deux réseaux de neurones profonds développés étant capables de localiser des organes et de classifier le nombre de feuilles des plantes, cette section propose de comparer leurs performances sur ces deux tâches. Afin de garantir une comparaison équitable, les deux réseaux sont entraînés sur la même base de données DB_2 avec la même définition d'image d'entrée de 512 $\times$ 512 et ils sont évalués sur la même base de validation. Les autres paramètres, comme la taille de tige, la taille du noyau gaussien de SDNet ou les seuils de confiance, sont fixés aux valeurs optimales établies dans la section 3.3.

La vitesse d'inférence calculée sur un NVIDIA Jetson Xavier de YT4 dans cette configuration est d'environ 22 ms (45 IPS) alors que celle de SDNet est de 45 ms (22 IPS). YT4 est donc près de deux fois plus rapide que SDNet, mais il faut cependant garder à l'esprit que si les architectures et configurations sont comparables, le niveau d'optimisation de Darknet (le framework qui implémente YT4) est bien supérieur au niveau d'optimisation de PyTorch (utilisé pour implémenter SDNet) qui a pour objectif de simplifier la recherche et le développement mais ne délivre pas nécessairement les performances optimales. Ainsi, Darknet utilise des optimisations spécifiques aux cartes graphiques NVIDIA comme CUDNN, TensorRT, l'entraînement en précision mixte et l'inférence en virgule flottante demi-précision (FP16). Ces optimisations ne sont pas automatiquement disponibles avec PyTorch et n'ont pas été implémentées pour SDNet car elles requièrent une ingénierie supplémentaire n'étant pas l'objectif de ce travail de recherche. La vitesse d'inférence de SDNet reste cependant suffisante pour une application en temps réel et peut être optimisée si besoin.

En complément des résultats présentés dans cette section, les résultats en détection de plantes d'intérêt mesurés par le consortium Operose lors des deux dernières campagnes d'évaluations du Challenge ROSE sont détaillés dans l'annexe G. Le réseau implanté dans BIPBIP lors de ces évaluations est YOLOv4 Tiny.

3.4.1.1 Tâche de localisation d'organes

Les deux réseaux sont capables de détecter trois types d'organes différents sur la base de données DB_2 : les tiges de maïs, de haricot et les feuilles. Le tableau 3.9 compare les

Réseau	Label	Rappel	Précision	Score F1	MAE_{loc}
YT4	Haricot	83,5 ± 2,4 %	87,6 ± 2,2 %	85,5 %	**4,7** ± 0,26 mm
	Maïs	**94,6** ± 1,6 %	91,5 ± 2,0 %	93,0 %	**3,6** ± 0,19 mm
	Feuille	83,8 ± 1,2 %	88,7 ± 1,0 %	86,2 %	2,8 ± 0,07 mm
	Total	85,3 ± 1,0 %	88,9 ± 1,0 %	87,1 %	**3,2** ± 0,07 mm
SDNet	Haricot	**85,7** ± 2,4 %	**92,3** ± 1,6 %	**88,8 %**	5,1 ± 0,28 mm
	Maïs	93,1 ± 2,0 %	**96,0** ± 1,1 %	**94,5 %**	3,7 ± 0,19 mm
	Feuille	**86,1** ± 1,1 %	**91,8** ± 0,8 %	**88,8 %**	2,8 ± 0,08 mm
	Total	**87,0** ± 0,9 %	**92,4** ± 0,6 %	**89,6 %**	3,3 ± 0,08 mm

Tableau 3.9 : Comparaison de YOLOv4 Tiny et SDNet sur la tâche de localisation de points d'intérêt. Les résultats sont détaillés par type de point d'intérêt et les meilleurs résultats sont illustrés en gras. Les hyperparamètres des deux réseaux sont fixés aux valeurs optimales déduites précédemment.

performances de SDNet et de YT4 par type d'organe (variante Y_{kp}, voir section 3.2.2.1) avec les métriques de localisation de points d'intérêt. Les erreurs standard sont spécifiées pour chaque métrique.

Ces résultats montrent que SDNet produit un Rappel et une Précision supérieurs à YOLOv4 Tiny pour pratiquement tous les types de point d'intérêt, seul le Rappel sur le maïs étant inférieur (-1,5 %). Tous types de point confondus, SDNet produit un Rappel 1,7 % supérieur et une Précision 3,5 % supérieure, ce qui donne un Score F1 2,5 % plus élevé que YOLO. SDNet génère donc à la fois moins de faux positifs et moins d'oublis. Les erreurs standard montrent en revanche que les résultats par type de point d'intérêts sur le Rappel et la Précision sont modérément significatifs. La base de données gagnerait à être plus grande afin de réduire les marges d'erreurs. L'erreur standard sur la Précision totale montre néanmoins que le gain en Précision peut être considéré comme significatif. L'erreur de localisation évaluée par le MAE_{loc} montre des performances légèrement supérieures pour YT4, mais au regard des erreurs standard les différences de performance ne sont pas significatives. Les deux réseaux commettent une erreur en localisation d'environ 3,2 mm tous types de points d'intérêt confondus.

La figure 3.16 illustre des prédictions réussies de points d'intérêt d'organes de plantes sur des plants de haricot (ligne du haut) et de maïs (ligne du bas) avec les tiges de haricot en bleu foncé, les tiges de maïs en bleu clair et les feuilles en violet (les résultats sont similaires pour YT4). Elle montre que SDNet est capable de détecter précisément les petites feuilles secondaires du haricot (en haut à gauche) ainsi que des feuilles de haricot à des stades plus avancés (en haut à droite). Les prédictions sur la ligne du bas montrent que SDNet est capable de détecter les feuilles de maïs dans des positions inhabituelles (à gauche) et est capable de localiser les organes même lorsque les plantes sont recouvertes par les adventices (à droite).

Cependant, si les résultats de SDNet sont globalement meilleurs que ceux de YT4,

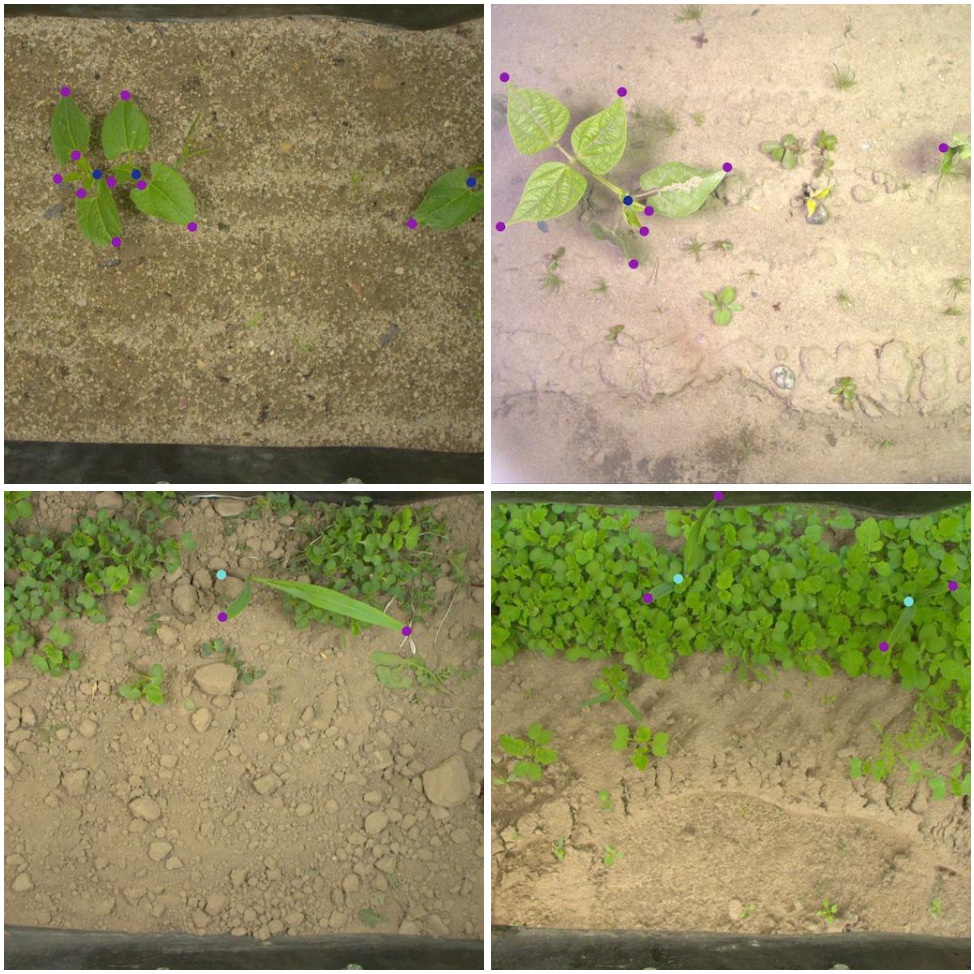


FIGURE 3.16 : Illustrations de détections réussies de points d'intérêt d'organes de plantes par SDNet. Les tiges de maïs sont représentées en bleu clair, les tiges de haricot en bleu foncé et les feuilles en violet.

il faut souligner que les deux réseaux présentent le même comportement vis-à-vis des différences de performance selon le type de point d'intérêt. Par exemple pour SDNet, les tiges de haricot sont moins bien détectées et moins bien localisées que les tiges de maïs : le Score F1 est inférieur de -5,7 % par rapport à celui du maïs et le MAE_{loc} est supérieur de plus de 1 mm. Ce phénomène traduit une plus grande difficulté de détection de la tige de haricot par rapport à celle du maïs intrinsèque au jeu de données. En effet, le motif de semis des deux plantes étudiées présentées dans la figure 1.2 de la section 1.1 montre que le semis du haricot est significativement plus serré que celui du maïs. Cet espacement plus faible induit une plus forte densité végétale qui peut être la source de recouvrement entre plantes adjacentes et peut causer des confusions et des difficultés de détection des tiges de haricot.

Les feuilles souffrent aussi d'une moins bonne détection, le score F1 étant plus ou moins équivalent à celui de la tige de haricot pour les deux architectures neuronales. En revanche, le MAE_{loc} est le plus bas des trois points d'intérêt étudiés : il est environ 1 mm inférieur à celui de la tige de maïs et 2,5 mm inférieur à celui de la tige de haricot concernant SDNet.

FIGURE 3.17 : Illustrations de détections erronées de points d'intérêt d'organes de plantes par SDNet. Les tiges de maïs sont représentées en bleu clair, les tiges de haricot en bleu foncé et les feuilles en violet.

Ce meilleur score de localisation malgré la moins bonne détection traduit une plus grande facilité des réseaux pour localiser précisément la position de la feuille. En effet, la position de la feuille est définie dans notre étude par la pointe de celle-ci. Ce point d'intérêt est plus texturé et plus caractérisable que les points d'intérêt de tige par son aspect visuel plus singulier et très localisé, ce qui rend sa localisation plus aisée. De plus, les tiges de plantes sont plus souvent masquées par les feuilles (de la plante, des plantes adjacentes ou des adventices) que les pointes de feuilles, ce qui rend les tiges plus difficiles à localiser précisément.

La figure 3.17 illustre des cas de détections erronées de points d'intérêt réalisées par SDNet (les erreurs de prédiction de YT4 sont similaires). Certaines adventices ressemblant fortement à des haricots occasionnent parfois des faux positifs. Par exemple, sur l'image de haricot en haut à gauche et sur l'image sans plante en bas à droite, trois feuilles d'une adventice sont détectées à tort à cause de la forte ressemblance avec les feuilles de haricot. Une autre erreur commise est la double détection de feuilles lorsque celles-si sont malformées. Par exemple, sur l'image en haut à droite un haricot présente une feuille

Réseau	Rappel	Précision	Score F1	MAE_{count}
YOLOv4 Tiny	56,2 ± 2,4 %	**70,5** ± 2,4 %	62,6 %	**0,3** ± 0,03 feuilles
SDNet	**63,0** ± 2,3 %	66,5 ± 2,3 %	**64,7 %**	0,4 ± 0,03 feuilles

Tableau 3.10 : Comparaison de YOLOv4 Tiny et SDNet sur la tâche d'estimation du nombre de feuilles, tous types de plantes et de nombre de feuilles confondus.

coupée en deux, causant un faux positif. Finalement, les prédictions sur le maïs présentées sur l'image en bas à gauche montrent une erreur de double détection de tige (maïs de gauche). Puisque la base des tiges est souvent masquée par le haut de la plante lorsque celle-ci est verticale, le réseau a tendance à prendre ce point comme référence[20]. Parfois le réseau génère deux détections pour la même tige : une à la base et une au sommet de la plante, causant un faux positif.

3.4.1.2 Tâche d'estimation du nombre de feuilles

La tâche d'estimation du nombre de feuilles est par nature plus difficile : il s'agit à la fois de localiser les plantes, d'inférer la variété de la culture et d'estimer le nombre de feuilles correct. La base de données DB_2 sur laquelle cette tâche est évaluée contient des plantes ayant un nombre maximal de feuilles de 8 donc YT4 est entraîné pour estimer 8 feuilles au maximum (configuration Y_{cl}, voir section 3.2.2.1) et les deux réseaux sont également évalués sur ce nombre maximal de feuilles. Les résultats sont présentés de manière synthétique dans le tableau 3.10 puis détaillés par type de plante (maïs et haricot), par nombre de feuilles et par réseau (SDNet et YOLOv4 Tiny) dans la figure 3.18.

Le tableau 3.10 détaille la performance tous types de plante et de nombre de feuilles confondus. On remarque d'abord que le compromis Rappel-Précision est différent pour les deux architectures : SDNet a un Rappel 6,80 % supérieur à YOLO tandis que ce dernier a une Précision 3,94 % plus élevée. SDNet produit donc moins d'oublis lors de la détection et de l'estimation du nombre de feuilles mais génère en revanche plus de faux positifs. En d'autres termes, SDNet permet de détecter et de classifier correctement plus de plantes mais il se trompe plus souvent sur le nombre de feuilles que YOLOv4 Tiny. Au regard des erreurs standard, les résultats sur la Précision sont modérément significatif et ceux sur le Rappel sont significatifs. Le Score F1 synthétisant ces deux indicateurs montre un avantage pour SDNet qui produit un score 2,18 % meilleur que YOLOv4 Tiny. L'erreur moyenne absolue MAE_{count} dans la régression du nombre de feuilles est équivalente pour les deux réseaux de neurones et est d'environ 0,3 feuilles. Cela montre que lorsque la régression du nombre de feuilles est erronée, celle-ci reste faible et dépasse rarement la feuille en trop ou en moins.

[20]Lorsque la base de la tige n'est pas visible sur une plante, la position la plus probable de la base est annotée afin que le réseau apprenne lui aussi à estimer ce point d'intérêt. Le haut de la plante est donc souvent annoté, ce qui peut induire en erreur le réseau.

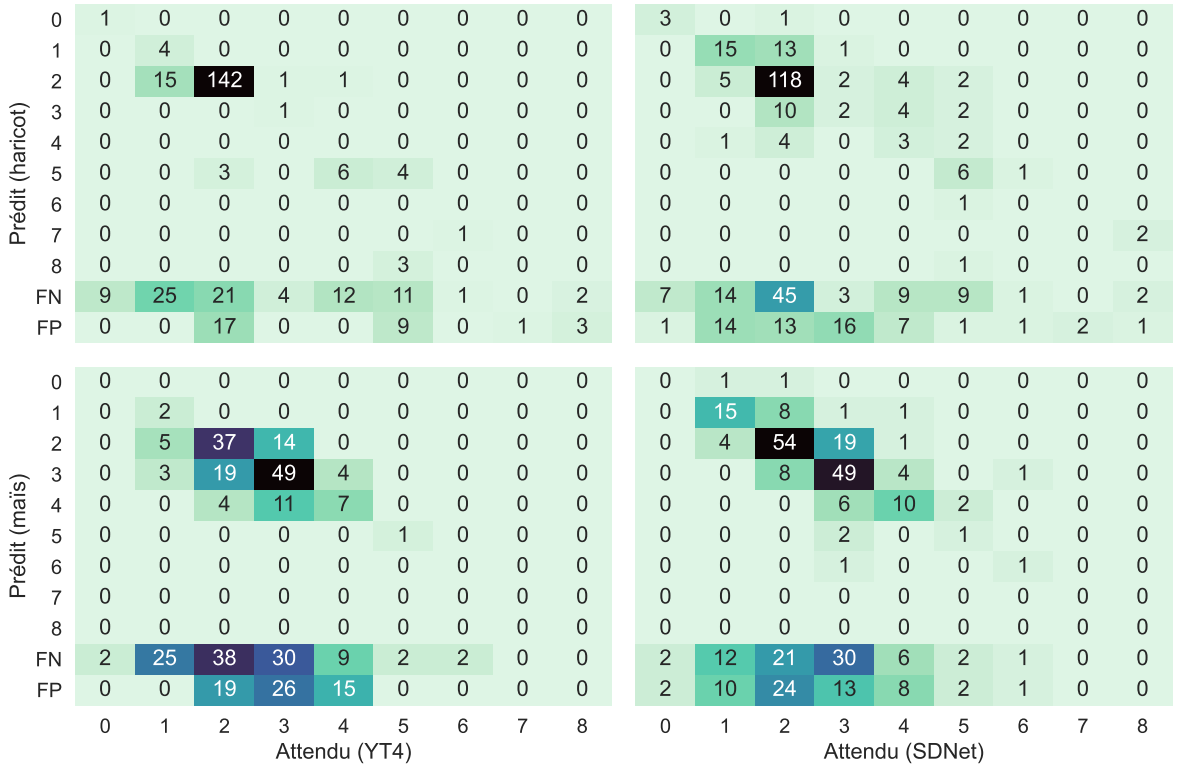


FIGURE 3.18 : Matrices de confusion sur le nombre de feuilles prédites sur la tâche d'estimation du nombre de feuilles pour les deux architectures neuronales. YOLOv4 Tiny (colonne de gauche), SDNet (colonne de droite), haricot (en haut) et maïs (en bas). Les indices correspondent au nombre de feuilles attendu et au nombre de feuilles prédit et les valeurs des cellules au nombre de représentants. Les lignes supplémentaires FN et FP réfèrent aux erreurs de détection, respectivement les faux négatifs (oublis) et les faux positifs.

La figure 3.18 présente les matrices de confusion sur l'estimation du nombre de feuilles pour les deux réseaux et les deux types de plantes étudiées. Il faut noter que le nombre de représentants de chaque catégorie (nombre de feuilles) est faible à cause de la petite taille de la base de données et de la fragmentation par type de plante et par nombre de feuilles. Ces résultats sont donc peu significatifs et donnés à titre informatif. Le nombre de feuilles attendu est représenté en abscisse et le nombre prédit en ordonnée. Le nombre de faux positifs (détections prédites à la mauvaise localisation, en doublon, ou avec le mauvais label de plante) et de faux négatifs (oublis) sont aussi représentés. Cette figure fait ressortir la distribution inégale du nombre de feuilles par plante (détaillée en annexe E). Ces matrices montrent que les deux réseaux atteignent des performances similaires en estimation du nombre de feuilles mais que la répartition n'est pas la même selon le type de plante.

Sur le haricot (ligne du haut), SDNet produit moins de TP que YOLOv4 Tiny sur les haricots à deux feuilles (-24) et les erreurs d'estimation sont principalement distribuées sur le haricot à une et trois feuilles (23 éléments), soit une erreur absolue d'une feuille.

En revanche, SDNet est meilleur sur les autres nombres de feuilles : +2, +11 et +6 respectivement pour le haricot à zéro feuille, une feuille, et trois feuilles et plus. L'erreur la plus courante de YOLOv4 Tiny est la surestimation du nombre de feuilles (de 1 feuille), fréquemment observée pour le haricot à une (15 surestimations) et quatre feuilles (6 surestimations). Sur le haricot, SDNet produit donc globalement plus d'erreurs d'estimation que YOLOv4 Tiny et celles-ci sont principalement situées sur le haricot à deux feuilles (la classe la plus représentée). SDNet est en revanche meilleur sur le reste de la plage de valeur.

Sur le maïs, SDNet prédit plus de TP que YOLOv4 Tiny sur l'ensemble de la plage : +13, +17, +4 pour le maïs à une feuille, deux feuilles, et quatre feuilles et plus. Le score est en revanche identique pour le maïs à trois feuilles (49 éléments). Les erreurs d'estimation de YOLOv4 Tiny sont principalement des surestimations du nombre de feuilles (de +1 ou +2 feuilles) : 8 surestimations pour le maïs à une feuille et 23 surestimations pour le maïs à deux feuilles. Sur le maïs SDNet est donc globalement meilleur quel que soit le nombre de feuilles.

Globalement SDNet est moins performant que YOLOv4 Tiny sur le haricot mais meilleur sur le maïs. YOLOv4 Tiny a tendance à surestimer le nombre de feuilles des plantes quelle que soit la variété alors que l'erreur de SDNet est répartie plus uniformément. SDNet retrouve plus de TP que YOLOv4 Tiny mais génère, dans le même temps, plus de faux positifs (erreur d'estimation du nombre de feuilles), ce qui illustre le meilleur score de Rappel et la moins bonne Précision de SDNet évoqués dans le tableau 3.10. On remarque aussi que SDNet est meilleur sur une plus grande plage de valeurs du nombre de feuilles, et notamment sur les petits nombres (haricot et maïs à une feuille).

La figure 3.19 illustre quelques exemples d'estimations erronées du nombre de feuilles et des erreurs de détection de YT4. Deux exemples sur le haricot sont présentés sur la ligne du haut et deux exemples sur le maïs sont présentés sur la ligne du bas. L'image de haricot de gauche montre deux oublis de détection (FN) et l'image de droite montre une double détection d'un plant, causant une fausse détection (FP). La première image de maïs à gauche montre une surestimation du nombre de feuilles (maïs à gauche) qui est estimée à 6 feuilles au lieu de 3. L'image montre au contraire une sous-estimation du nombre de feuilles (2 au lieu de 3), certainement causée par le masquage d'une feuille par une autre (elles sont presque superposées).

La figure 3.20 illustre des exemples d'estimation erronée du nombre de feuilles et des erreurs de détection de SDNet. L'image de haricot en haut à gauche montre des détections dupliquées de feuilles (haricot de gauche) conduisant à une estimation de 4 feuilles au lieu de 2 attendues. L'image de haricot de droite montre une fausse détection de feuille sur une adventice menant à une surestimation d'une feuille (haricot de gauche). L'image sur le maïs en bas à gauche montre un oubli de détection (FN) causé par la forte proximité des deux plantes (et la forte parallaxe qui a provoqué une superposition des bases de tige)

FIGURE 3.19 : Illustrations d'estimations erronées du nombre de feuilles par YT4. La boîte englobante des plantes est présentée avec un label indiquant le type de plante et le nombre de feuilles.

et qui a provoqué une association erronée des feuilles d'une plante à l'autre et donc à une surestimation du nombre de feuilles. Finalement, l'exemple en bas à droite montre une double détection de la base de la tige causée par une confusion entre la base et le haut de la plante (voir la section précédente) et occasionnant une fausse détection (FP) et une sous-estimation du nombre de feuilles (il manque une feuille à la plante correctement détectée puisqu'elle a été associée à la fausse détection).

Pour finir, la figure 3.21 illustre des détections et estimations du nombre de feuilles réussies pour YT4 (à gauche) et SDNet (à droite). Les prédictions de YT4 (figure 3.21a) illustrent des détections de jeunes plantes à une feuille (boîtes roses) et des estimations réussies du nombre de feuilles malgré une infestation d'adventices (image en bas à droite). Les prédictions de SDNet (figure 3.21b) illustrent des estimations correctes du nombre de feuilles et de la position des organes malgré une grande envergure des plantes (image en bas à gauche) et un stade de développement avancé (image en haut à droite).

FIGURE 3.20 : Illustrations d'estimations erronées du nombre de feuilles par SDNet. La structure est annotée avec la base de tige de maïs en bleu clair, celle du haricot en bleu foncé et les feuilles en violet reliées par un segment blanc à l'ancre correspondante.

3.4.2 Performance de l'agrégation temporelle

Cette section quantifie l'apport de performance sur la détection et la localisation des tiges de plantes de l'algorithme d'agrégation temporelle. La base de données de séquences d'images est utilisée et les métriques de localisation de points d'intérêt sont employées pour comparer les performances. Les deux réseaux de neurones peuvent être utilisés pour prédire les tiges de plantes nécessaires à l'algorithme d'agrégation, ils sont donc aussi comparés. Comme dans les deux sections précédentes, SDNet et YT4 sont entraînés sur la base de données DB_2 dans des conditions similaires et équitables (configuration Y_{kp} pour YT4). La base de données DB_3 n'étant pas très grande, les résultats présentés dans cette section illustrent des tendances.

La rapidité d'exécution de l'algorithme d'agrégation temporelle est dominée par le coût calculatoire du flot optique dense qui s'exécute à une vitesse d'une dizaine d'images par seconde sur un NVIDIA Jetson Xavier. Les autres composants (calcul des masques de la végétation, projection des détections, algorithme d'association) sont largement temps-

(a) YOLOv4 Tiny (b) SDNet

FIGURE 3.21 : Illustrations d'estimations réussies du nombre de feuilles par YT4 (à gauche) et SDNet (à droite). Les deux réseaux sont capables de détecter de très jeunes plantes (boîtes roses de maïs à une feuille sur la ligne du bas de la figure 3.21a) et des plantes de grande envergure ou plus matures (haricot en haut à droite et maïs en bas à gauche sur la figure 3.21b).

Label	Agr.	Rappel	Précision	Score F1	MAE_{loc}
Haricot	sans	75,3 %	**96,6 %**	84,6 %	**5,47** ± 0,27 mm
	avec	**82,5 %**	94,8 %	**88,2 %**	5,62 ± 0,23 mm
Maïs	sans	75,7 %	95,6 %	84,5 %	**4,71** ± 0,34 mm
	avec	**95,1 %**	**95,8 %**	**95,5 %**	6,92 ± 0,34 mm
Total	sans	75,5 %	**96,1 %**	84,6 %	**5,09** ± 0,30 mm
	avec	**88,8 %**	95,3 %	**91,8 %**	6,27 ± 0,29 mm

TABLEAU 3.11 : Apports de performance de l'algorithme d'agrégation temporelle sur les prédictions de YT4. Les meilleures performances sont illustrées en gras.

réel (supérieur à 100 IPS). Comme pour SDNet, l'algorithme d'agrégation n'a pas fait l'objet d'un travail d'ingénierie visant à optimiser l'exécution sur du matériel embarqué. Néanmoins, la section 3.5 présente quelques perspectives simples à mettre en œuvre pour accélérer le calcul du flot optique au besoin.

Le tableau 3.11 présente les performances de l'algorithme d'agrégation lorsque YT4 est utilisé comme générateur de prédictions de tiges. Les résultats sont comparés par type de tige (haricot et maïs), avec et sans agrégation (Agr.). De manière synthétique (ligne "Total" tous types de tiges confondus), l'algorithme d'agrégation améliore considérablement le Rappel qui augmente de 13,3 % mais diminue légèrement la Précision qui chute de 0.8 %. Le Score F1, synthétisant ces deux indicateurs, bénéficie d'une augmentation de 7,2 %. Le score de localisation MAE_{loc} quant à lui subit une légère baisse de 1,18 mm, modérément

Label	Agr.	Rappel	Précision	Score F1	MAE_{loc}
Haricot	sans	89,4 %	**95,9 %**	92,5 %	**5,05** ± 0,27 mm
	avec	**92,0 %**	93,4 %	**92,7 %**	5,36 ± 0,23 mm
Maïs	sans	85,4 %	**96,1 %**	90,4 %	**4,83** ± 0,34 mm
	avec	**95,1 %**	95,8 %	**95,5 %**	6,73 ± 0,34 mm
Total	sans	87,4 %	**96,0 %**	91,5 %	**4,94** ± 0,30 mm
	avec	**93,6 %**	94,6 %	**94,1 %**	6,04 ± 0,29 mm

TABLEAU 3.12 : Apports de performance de l'algorithme d'agrégation temporelle sur les prédictions de SDNet. Les meilleures performances sont illustrées en gras.

significative d'après l'erreur standard fournie. Ce comportement global est en fait le même pour les deux types de tiges : le Rappel profite d'une forte augmentation (+7,2 % pour le haricot et +19,4 % pour le maïs) et la Précision diminue légèrement ou reste stable (-1,8 % pour le haricot et +0,2 % pour le maïs), ce qui se traduit dans les deux cas par un Score F1 augmentant significativement (+3,6 % pour le haricot et +11,0 % pour le maïs). Le MAE_{loc} diminue dans les deux cas mais c'est principalement le maïs qui perd en précision de localisation : -0,15 mm pour le haricot et -2,21 mm pour le maïs. Il faut cependant noter que le gain de performance est meilleur pour le maïs par rapport au haricot.

Le tableau 3.12 présente les performances de l'algorithme d'agrégation temporelle lorsque SDNet est employé en tant que générateur de prédictions de tiges. Il dresse un constat similaire à celui fait pour YT4 : l'algorithme d'agrégation améliore fortement le Rappel quel que soit le type de tige mais il diminue la Précision et dégrade légèrement la précision en localisation mesurée par le MAE_{loc}. Au total (tous types de tiges confondus), le Rappel est amélioré de 6,2 %, la Précision diminue légèrement de 1,4 %, le Score F1 augmente de 2,6 % et le MAE_{loc} augmente de 1,1 mm.

Globalement, l'apport de performance est similaire quel que soit le générateur de prédictions de tiges employé et quel que soit le type de tige. L'algorithme d'agrégation élimine une grande partie des oublis (FN) produits par les réseaux de détection au prix d'une légère augmentation des faux positifs (FP) et d'une petite baisse en précision de localisation. Le fort gain en Rappel est une conséquence directe du principe de fonctionnement de l'algorithme : observer une plante plusieurs fois sous différents profils donne plus de chances de la détecter correctement au moins une fois. En revanche, dans le même temps, le processus d'agrégation donne plus d'opportunités de conserver un faux positif (une adventice par exemple) qui apparaîtrait plusieurs fois dans la séquence, ce qui explique la baisse de Précision. La diminution de la précision en localisation peut être expliquée quant à elle par deux facteurs :

- D'abord, les hypothèses réalisées sur le mouvement et la position de la caméra peuvent ne pas être parfaitement respectées. Par exemple, le module vision peut s'incliner à cause d'une bosse sur le sol ou la hauteur de la caméra par rapport au sol peut diminuer si le sol est plus meuble. Ceci peut se répercuter sur la qualité du

flot optique et fausser la mesure de l'avancement de la caméra et finalement affecter la qualité de la projection des détections dans le référentiel commun (voir section 2.4) et donc générer des imprécisions de localisation.

- D'autre part, la forte parallaxe due à la proximité de la caméra avec le sol cause de nombreux masquages de la base de la tige par d'autres plantes ou adventices. Dans ces cas, la prédiction des réseaux peut être erronée et la tige localisée au mauvais endroit. Ce phénomène est d'autant plus important pour les grands plants de maïs qui souffrent du syndrome de la double détection : parfois le sommet de la plante est détecté en même temps que la base de la tige (voir les sections 3.4.1.1 et 3.4.1.2), causant nécessairement une plus forte erreur de localisation. Ceci peut expliquer la plus forte perte de performance en localisation sur le maïs (-1,9 mm) que sur le haricot (-0,3 mm) pour SDNet.

L'apport de performance plus important sur le maïs que sur le haricot évoqué plus haut peut quant à lui s'expliquer par leur schéma de semis respectifs différent. La distance moyenne entre les plantes d_{within} est en effet plus grande pour le maïs (15 cm) que pour le haricot (entre 3 cm et 8 cm). Pour le haricot il y a donc plus de possibilités de recouvrement entre plantes ou de confusion sur la position de la tige. Ce schéma de semis plus serré peut donc causer plus de difficultés à l'algorithme d'agrégation temporelle sur le haricot.

La figure 3.22 illustre des prédictions filtrées par l'algorithme d'agrégation lorsque YT4 est utilisé pour la détection de tiges sur un extrait de séquence de maïs (ligne du haut) et un extrait de séquence de haricot (ligne du bas). Les trackers sont représentés par les amas de points d'une même couleur où chaque point représente une détection de tige associée au tracker. Les ellipses de dispersion [119] à 1 σ (contenant environ 68 % des échantillons) sont aussi présentées pour chaque tracker. La séquence sur le maïs met en lumière le phénomène de double détection de la tige et l'incertitude sur la position exacte causée par les masquages. Par exemple, sur la première image le tracker jaune a une forme allongée qui est causée par des détections estimées au sommet de la plante plutôt qu'à la base de la tige. La feuille droite de la plante, orientée dans l'axe d'avancement du module, a pu masquer la tige lors des observations consécutives. La troisième image sur le maïs montre un faux positif (tracker jaune) causé par plusieurs détections erronées réalisées sur une feuille au lieu de la tige. La séquence sur le haricot montre un faux positif causé par plusieurs détections erronées d'une adventice (image du milieu) et un oubli de détection causé par un masquage de la tige par la feuille d'une adventice (haricot au-dessus à gauche du tracker orange). La section 3.5 discute des possibilités de post-traitement qui pourraient être conduites sur ces nuages de points.

Les résultats présentés dans les deux tableaux permettent aussi de comparer les performances des deux générateurs de détection de tige, avec et sans l'algorithme d'agrégation temporelle. SDNet conserve son avantage sur YT4 sur cette base de données : son Score F1

FIGURE 3.22 : Illustration des trackers produits par l'agrégation temporelle sur deux séquences d'images. Les détections d'un même tracker ainsi que l'ellipse de dispersion à 1 σ (contenant 68% des échantillons) des détections sont représentées avec la même couleur.

est 6,9 % supérieur (tous types de tiges confondus) sans l'algorithme d'agrégation et 2,3 % supérieur avec l'algorithme d'agrégation. Le gain de performance est dû à un meilleur Rappel (+11,9 % dans le cas sans agrégation), la Précision étant équivalente (environ 96 %). Le score de localisation $\mathrm{MAE}_{\mathrm{loc}}$ est légèrement meilleur pour SDNet (-0.15 mm sans agrégation, -0,23 mm avec) sans que cela soit significatif au regard de l'erreur standard. En revanche, les résultats par variété de tige sont différents de ceux obtenus dans la partie 3.4.1.1 : le haricot est un peu mieux détecté que le maïs (sans l'agrégation). Une des causes de cette divergence est la variation des conditions expérimentales. En effet, les séquences sur le maïs sont réalisées sur des planches de culture à des stades de développement plus avancés que la base de données DB_2, le maïs est donc plus difficile à détecter sur celle-ci. Les résultats montrent cependant que l'algorithme d'agrégation renverse cette tendance, le Score F1 du maïs passant au-dessus de celui du haricot (+2,8 % pour SDNet par exemple).

3.5 Synthèse et discussion

Après avoir introduit les tâches évaluées, les métriques et les bases de données employées, ce chapitre a présenté des expérimentations sur le paramétrage des algorithmes et une comparaison des performances.

La section 3.3 qui détaille le paramétrage des algorithmes via des recherches exhaustives montre que les réseaux de neurones profonds sont flexibles et remarquablement robustes. Utiliser un réseau de détection d'objets comme détecteur de points d'intérêt s'est révélé concluant et cette approche rivalise avec les performances d'un réseau de neurones dédié (SDNet). Les expérimentations sur la taille de la boîte englobante de tige indiquent que ce paramètre a une faible influence sur la performance et montrent que les réseaux de neurones sont en général peu sensibles aux problèmes "mal posés" ou utilisant des modélisations peu communes, comme des boîtes englobantes de tailles arbitraires pour modéliser des points d'intérêt. Cette section montre aussi une conclusion classique du domaine de l'apprentissage profond, à l'origine même de son nom : en général plus un réseau est profond, plus il est performant. Cependant, ce gain de performance est contrebalancé par la perte en vitesse d'exécution (et d'entraînement) causée par l'augmentation du nombre d'opérations inhérente à l'augmentation de la profondeur : c'est ce que l'on appelle le compromis vitesse-précision. Les résultats présentés montrent aussi que d'autres paramètres, comme les seuils de confiance, contrôlent un compromis entre le Rappel et la Précision. Il est donc possible de favoriser l'une de ces métriques en fonction de l'objectif visé. Par exemple, déraciner une plante d'intérêt diminue irrémédiablement le rendement alors que rater une adventice a moins d'impact et un second passage du module de désherbage peut rattraper l'erreur. Dans ce cas, favoriser le Rappel peut être stratégique afin de maximiser le rendement. Finalement, au travers du paramétrage de l'algorithme d'agrégation, cette section montre que la spécificité de certaines cultures (l'espacement d_{within} entre les plantes par exemple) a une influence directe sur la performance des algorithmes. Il est donc nécessaire de les prendre en compte dans le paramétrage pour optimiser la performance.

Sur un autre registre, cette section montre que si les recherches exhaustives permettent effectivement de trouver un paramétrage optimal et d'augmenter les performances, celles-ci sont généralement très coûteuses en temps de calcul malgré la profondeur relative des réseaux employés dans notre cas. Les réseaux employés apparaissent en effet peu profonds lorsqu'ils sont comparés aux réseaux "state of the art" à plusieurs centaines de milliards de paramètres (comme GPT-3 [20]) sur lesquels les recherches exhaustives ne sont pas envisagées : un entraînement dure en général plusieurs semaines sur un serveur de calcul distribué contenant plusieurs dizaines de cartes graphiques et coûte en général de plusieurs dizaines à plusieurs centaines de milliers de dollars [124]. Ces architectures très profondes nécessitent aussi de très grandes quantités de données d'entraînement, ce qui n'est pas évident pour des tâches aussi spécialisées que le désherbage à un stade précoce. Ce constat pose la question de l'*explicabilité* des réseaux profonds et de la faiblesse des moyens disponibles pour comprendre l'origine des performances. Les réseaux profonds sont des *boîtes noires* dont l'interprétation des résultats est difficile et coûteuse. Les recherches exhaustives présentent aussi un autre risque, exacerbé par la petite taille des bases de données employées : le risque de sur-apprentissage (*over-fitting*). Les valeurs des

paramètres trouvées par les recherches exhaustives sont des optimums locaux et propres aux données de validation, il est donc possible que sur des données issues de conditions réelles différentes du jeu de validation (*domain shift*), ces paramètres ne permettent pas d'obtenir les performances optimales.

La section 3.4 présentant les comparaisons des performances des algorithmes montre que l'architecture neuronale dédiée SDNet permet un gain de performance par rapport à l'utilisation d'un détecteur d'objet pour réaliser les tâches de localisation d'organes de plantes et l'estimation du nombre de feuilles. À l'inverse de ce dernier, SDNet n'a besoin que d'un seul entraînement pour pouvoir réaliser les deux tâches, et il se révèle plus flexible : il est capable de détecter un nombre arbitraire de feuilles et donc d'estimer un nombre arbitraire de feuilles pour chaque plante. SDNet est aussi plus facilement explicable. Par exemple, une erreur dans l'estimation du nombre de feuilles peut être expliquée par une détection manquante de feuille ou par l'association d'une feuille d'une autre plante (causée par un masquage par exemple). Ceci permet éventuellement d'améliorer la base de données avec des images prises dans les conditions problématiques soulevées par les erreurs les plus fréquentes. Cette section montre aussi qu'un algorithme d'agrégation temporelle simple permet d'améliorer sensiblement le Rappel (i.e. diminuer le nombre d'oublis) de la détection de tige dans le cas du bloc-outil BIPBIP en tirant parti de la redondance d'information dans les séquences d'images. La vitesse d'exécution de l'algorithme d'agrégation est cependant limitée par le calcul du flot optique dense qui est relativement lent car le calcul du flot est réalisé pour toutes les positions de l'image (i.e. à chaque pixel). Une perspective d'amélioration est l'emploi d'un algorithme de calcul du flot optique non dense (*sparse*) ou le flot optique n'est calculé qu'à certaines positions prédéfinies. De tels algorithmes existent, comme le traqueur de points d'intérêts Lucas-Kanade pyramidal [17], implémenté dans la librairie OpenCV. Afin d'augmenter davantage la vitesse d'exécution, l'algorithme peut aussi être exécuté sur un GPU. Un autre angle d'amélioration est la perspective de filtrage supplémentaire des nuages de points illustrée dans la figure 3.22 présentant des exemples d'ellipses de confiance de trackers. Il pourrait être judicieux de tenir compte d'a priori sur le semis, comme l'espacement régulier entre les plantes et la ligne de culture. Détecter la ligne de culture permettrait de supprimer les prédictions étant en dehors d'une certaine zone de confiance centrée sur la ligne et qui ont plus de probabilité d'être de faux positifs (par exemple le tracker en jaune sur l'image en haut à gauche de la figure 3.22).

Cependant, les résultats de cette section montrent aussi la forte dépendance aux données des algorithmes d'apprentissage profond. La qualité des images et des annotations est en effet cruciale pour que les réseaux apprennent correctement. La qualité de la donnée est tellement importante que certains experts du domaine (Andrew Ng, docteur et co-fondateur de Google Brain par exemple) sont en faveur d'une approche de l'intelligence artificielle centrée sur la donnée (*Data-centric AI*) plutôt que sur le modèle (*Model-centric AI*) lorsqu'il s'agit d'améliorer les performances. En d'autres termes, il est probablement

FIGURE 3.23 : Illustration de l'importance de la qualité des données sur la performance des réseaux de neurones.

plus pertinent de collecter des données plus diversifiées et qui couvrent plus de situations "exceptionnelles" plutôt que d'améliorer un réseau pour gagner en performance. La figure 3.23 illustre des exemples de prédictions de YT4 et SDNet sur la base de données DB_2 (sur laquelle ils sont entraînés dans des conditions similaires) qui démontrent l'importance des données. La première ligne montre un cas où les deux réseaux réalisent exactement la même erreur : ils détectent un piquet en plastique jaune comme un maïs à une feuille. La base d'entraînement ne contient pas d'image avec des piquets jaunes et il est probable que les réseaux aient appris à détecter les tiges de maïs comme des objets longs et jaunes, d'où l'erreur de généralisation commise. La ligne du bas montre aussi une erreur de prédiction similaire pour les deux réseaux : deux plantes de haricot très proches sont détectées comme une seule plante. Les réseaux échouent de la même manière et il est probable que la base de données soit la cause de ces erreurs. Ajouter des images avec des piquets jaunes et des images avec des plants très serrés réduirait probablement la fréquence de telles erreurs. Une autre remarque connexe est la consistance des annotations, i.e. la variation de l'annotation entre plusieurs annotateurs. Non seulement une consigne d'annotation peut être très subjective ("annoter les tiges de plantes"), mais la tâche peut être suffisamment

difficile (par exemple estimer la position de la base de tige qui est rarement visible à cause des masquages) pour que les annotations divergent entre plusieurs annotateurs. Cette inconsistance des annotations affecte la qualité des données et donc la performance des algorithmes d'apprentissage profond.

La quantité des données est aussi importante que la qualité pour que les algorithmes d'apprentissage profond apprennent correctement et bénéficient d'une bonne capacité de généralisation et la tendance d'apprendre sur toujours plus de données ne semble pas faiblir. C'est par exemple le cas des architectures *Transformers* [24, 32] qui commencent à être utilisées pour les tâches de vision comme la classification et la détection d'objets et qui requièrent des quantités astronomiques de données. C'est pur cette raison que les bases de données de classification d'image et de détection d'objets comme ImageNet [30], COCO [86] et OpenImage [11, 75] contiennent un nombre très élevé d'images et d'annotations : plus de 1 million d'images pour ImageNet, plus de 300 milliers pour COCO et plus de 9 millions pour la version 6 du jeu de données OpenImage. La taille de nos bases de données sont faibles en comparaison : la plus grande (DB_1) possède un peu plus de 2 000 images. Collecter et annoter plus d'images à plus de stades de développement et dans plus de conditions différentes (couleur et granularité du sol, infestation d'adventices) est une nécessité pour le développement futur de réseaux de neurones performants. Cependant, annoter de telles quantités d'images est un travail chronophage et coûteux s'il est délégué. La conception et la maintenance des bases de données requiert donc des méthodes d'optimisation telles que la pré-annotation (par un réseau déjà entraîné sur une autre base de données par exemple).

Conclusions et perspectives

L'objectif de ces travaux était d'explorer des méthodes de vision par ordinateur et d'apprentissage profond pour assister des tâches de désherbage de précision et de surveillance de l'état des cultures. D'une part, des méthodes d'apprentissage profond ont été étudiées afin de localiser et de caractériser la structure des plantes. D'autre part, un algorithme d'agrégation temporelle de détections de plantes visant à améliorer la performance de la prédiction dans le cas du bloc-outil de désherbage mécanique de précision BIPBIP a été proposé.

La première contribution concerne le choix et le paramétrage judicieux d'un réseau de neurones profond de détection d'objets employé pour localiser les organes de plantes, comme les bases de tiges et les feuilles, ainsi que pour estimer le nombre de feuilles des plantes. Plusieurs familles de réseaux de détection d'objets ont été analysées puis une famille, YOLO, a été sélectionnée en se basant sur des considérations de vitesse d'exécution et de précision des prédictions. Des recherches exhaustives ont ensuite été menées sur les hyperparamètres principaux (définition du réseau, taille de la boîte englobante de points d'intérêt, etc.) afin d'établir les valeurs optimales.

La seconde contribution consiste en un réseau de neurones profond innovant, inspiré des détecteurs de poses d'objets, capable d'estimer la structure non-contrainte des plantes, c'est-à-dire la position des organes et les liens entre ces organes. Ce réseau est capable de plus de flexibilité que les détecteurs d'objets puisqu'il est capable de prédire un nombre arbitraire de feuilles par plante et ne requiert qu'un entraînement pour apprendre deux tâches : la localisation d'organes des plantes et l'estimation du nombre de feuilles. Des recherches exhaustives ont été menées sur ses principaux hyperparamètres afin d'estimer leur valeur optimale puis ses performances ont été comparées avec celle du détecteur d'objet.

La troisième contribution est un algorithme d'agrégation temporelle permettant d'améliorer la performance de la détection des plantes dans des séquences d'images. Dans le cadre du désherbage mécanique de précision visé par le bloc-outil BIPBIP, cet algorithme est employé pour suivre les bases des tiges des plantes. Des expérimentations ont été réalisées pour estimer la valeur optimale des hyperparamètres puis la performance de l'algorithme a été évaluée.

La dernière contribution, expérimentale, concerne l'acquisition et l'annotation de bases

de données d'images et de séquences d'images destinées à entraîner les réseaux de neurones profonds et à les évaluer.

Le premier résultat intéressant est que, bien qu'ils ne soient pas conçus pour de telles applications, les réseaux de neurones profonds de détection d'objets peuvent être employés pour réaliser de la détection de points d'intérêt et permettent d'obtenir de bonnes performances. Les expérimentations sur les réseaux de la famille YOLO montrent qu'ils sont robustes et performants. En effet, les performances dépendent peu des principaux hyperparamètres tels que la définition du réseau ou la taille de la boîte englobante des points d'intérêts. C'est la profondeur du réseau qui a le plus d'influence sur la performance, cependant, elle a un impact négatif sur la vitesse d'exécution. Le choix d'une architecture neuronale adaptée à une application donnée dépend donc de ce compromis et du matériel de calcul embarqué disponible.

Le deuxième résultat remarquable est qu'une architecture neuronale consacrée à la détection de structure de plantes permet d'obtenir d'excellentes performances tout en étant très flexible. En effet, le détecteur SDNet est capable de prédire des structures de plantes ayant un nombre arbitraire de feuilles. Cette méthode permet d'unifier les tâches de localisation d'organes et d'estimation du nombre de feuilles au sein d'une même architecture ne requérant qu'un seul entraînement. Cela facilite ainsi le déploiement pour des applications de désherbage de précision et de phénotypage en plein champ.

Le troisième résultat est qu'un algorithme de suivi temporel basé sur de simples données d'imagerie permet d'améliorer significativement la performance de la prédiction des bases de tiges de plantes dans le cas du bloc-outil de désherbage de précision BIPBIP. Le gain de performance provient de l'amélioration du Rappel indiquant que l'algorithme permet principalement de réduire le nombre d'oublis de détection.

Ces contributions méthodologiques on fait l'objet de deux communications en conférence et d'un article de revue. Une partie de la chaîne algorithmique est implantée dans le module BIPBIP et a été évaluée par le consortium Operose dans le cadre du Challenge ROSE. Le bloc-outil de désherbage mécanique BIPBIP a également été primé au salon SIVAL de 2021 (médaille d'argent).

Les travaux réalisés et résultats obtenus laissent entrevoir des perspectives diverses.

Tout d'abord, la discussion des résultats soulève un point important : la performance des réseaux de neurones est extrêmement dépendante de la quantité et de la qualité des données d'entraînement, c'est-à-dire des images et des annotations. La donnée est le nerf de la guerre des méthodes d'apprentissage profond supervisées. Déployer un réseau de neurones pour une application commerciale requiert une stratégie de collecte, de traitement et d'annotation de données coordonnée sur le long terme afin de garantir un fonctionnement optimal dans la majorité des conditions. Une perspective notoire en vue d'une industrialisation

et d'une commercialisation du bloc-outil BIPBIP est donc l'amélioration des bases de données afin de couvrir plus d'espèces de plantes, plus de stades de développement et plus de conditions. L'annotation des données étant chronophage, on peut aussi envisager de pré-annoter les images, par exemple en utilisant un réseau de neurones déjà entrainé. L'analyse de la variabilité de l'annotation entre plusieurs annotateurs est également un axe d'étude important. Cela permet, d'une part, d'estimer la "difficulté" du jeu de données (et donc d'estimer la performance théorique d'un expert humain) et, d'autre part, de calculer l'impact des erreurs d'annotation sur la performance de détection.

Une perspective également importante pour un futur déploiement est l'amélioration de la vitesse d'exécution des algorithmes et notamment de l'algorithme d'agrégation. L'optimisation des méthodes n'étant pas l'objectif du travail proposé dans ce manuscrit, la vitesse d'exécution de certains algorithmes pourrait être améliorée. Tout d'abord, la vitesse d'inférence de SDNet pourrait être améliorée en utilisant des optimisations comme la réduction de la précision numérique des calculs en virgule flottante sur 16 bits plutôt que 32 bits (l'impact sur la précision étant négligeable) ou comme l'utilisation de framework d'inférence de haute performance comme TensorRT[21]. Ensuite, la vitesse d'exécution de l'algorithme d'agrégation temporelle, dominée par le temps de calcul du flot optique, pourrait être améliorée en employant des méthodes de calcul du flot optique non dense, c'est-à-dire opérant sur un nombre réduit de points de l'image plutôt qu'à la position de tous les pixels.

L'amélioration de la robustesse de l'algorithme d'agrégation temporelle est aussi une perspective d'amélioration. L'analyse des performances présentée met en lumière la forte dépendance de l'algorithme d'agrégation aux conditions particulières dans lesquelles le bloc-outil BIPBIP opère. De nombreuses hypothèses ont été réalisées sur le déplacement et la position de la caméra, ce qui pourrait provoquer des imprécisions de modélisation et affecter négativement les performances comme le montre la baisse de la précision en localisation causée par l'agrégation. Afin de rendre l'approche plus robuste, il serait pertinent de pouvoir calculer certains paramètres comme la distance de la caméra au sol et son inclinaison afin de corriger la prédiction du déplacement.

Le détecteur de structure SDNet démontre qu'il est possible de prédire des graphes en étoile ayant un nombre arbitraire de sommets. Une perspective d'évolution intéressante est la régression de structures plus complexes comme des graphes ou des arbres. En effet, la formulation actuelle de SDNet ne se limite pas à la régression de graphes étoilés. Puisque SDNet est capable de prédire un ensemble de nœuds V, via les cartes de chaleur, ainsi qu'un ensemble d'arêtes E (couple de nœuds), via les indices de regroupement, il peut en théorie prédire un graphe $G = (V, E)$. Une adaptation minimale du décodeur permettrait donc de prédire un ensemble de graphes. Une application possible en proxidétection agricole est la détection de la structure du réseau de branches d'arbres et de plantes buissonnantes

[21] `https://developer.nvidia.com/tensorrt`

comme la vigne. Plus généralement, on pourrait imaginer des applications dans d'autres domaines comme la détection de réseaux routiers ou de réseaux vasculaires.

Annexes

Annexe A

État de l'art sur les méthodes d'apprentissage profond

L'apprentissage profond (Deep Learning, abrégé DL) de réseaux de neurones artificiels (ANN) désigne un ensemble de méthodes de Machine Learning, et plus généralement d'Intelligence Artificielle [125]. L'émergence des premiers ANN est plutôt ancienne (1943) mais leur potentiel n'a pas pu être exploité avant le début des années 2000 à 2010 faute de méthode efficace d'apprentissage [6]. Ils commencent à gagner en popularité au début des années 2000 avec la conception du premier réseau de neurones convolutif (CNN) par Yann Lecun qui obtient d'excellentes performances dans un problème de reconnaissance de chiffres manuscrits [81]. Puis, en 2012, Alex Krizhevsky publie un article présentant des performances considérables sur un problème de classification d'un peu plus d'un million d'images haute-définition dans mille catégories possibles en utilisant un réseau de neurones profond (DNN) [74], et marque le coup d'envoi du développement massif des méthodes de Deep Learning.

Les réseaux profonds ne se limitent pas à l'analyse d'images, ils sont utilisés dans pratiquement tous les domaines comme la traduction, la reconnaissance vocale, le traitement de la parole et même la production artistique [6]. Cependant, ce sont les réseaux de traitement d'images qui nous intéressent dans la suite, et plus spécifiquement les CNN. D'autres architectures existent mais ne seront pas évoquées, par exemple les réseaux récurrents (RNN), les réseaux GAN et l'apprentissage par renforcement (RL). Les réseaux de neurones peuvent être catégorisés dans deux groupes : les approches supervisées et les approches non supervisées. La première approche nécessite des données annotées alors que la seconde n'en a pas besoin. Seule la première catégorie va nous intéresser. Les approches non supervisées sont encore peu développées, moins performantes et peu adaptées à l'application visée dans ce manuscrit.

A.1 Principes et bases théoriques des réseaux de neurones artificiels

"If a car travels twice as fast as a bicycle and the bicycle is four times as fast as a walking man, then the car travels $2 \times 4 = 8$ times as fast as the man." — George F. Simmons

Les réseaux de neurones artificiels (ANN) sont inspirés par bio-mimétisme de l'organisation et du fonctionnement des neurones biologiques dans le cerveau. Un neurone artificiel, appelé *perceptron*, est connecté à plusieurs autres neurones en amont par le biais de synapses par lesquelles il reçoit des signaux d'entrée. En retour, il synthétise un signal de sortie envoyé à tous les autres neurones connectés en aval via son axone. La structure d'ANN la plus courante, et celle sur laquelle ces travaux sont basés, est le réseau de neurones multicouches (Multi-Layer Perceptron (MLP)). Les neurones sont groupés en couches successives, chaque couche étant connectée à tous les neurones de la précédente (on parle d'une couche *Fully Connected* (FC)). Cette formation est illustrée sur la figure A.1b et le fonctionnement individuel d'un neurone est illustré par la figure A.1a. Les réseaux de neurones CNN évoqués plus haut sont des cas particuliers de MLP, leur fonctionnement est similaire à ces derniers à la différence qu'ils sont bien mieux optimisés et convergent plus rapidement dans les applications où les données possèdent une cohérence spatiale ou temporelle. Le fonctionnement d'un neurone artificiel peut être formalisé mathématiquement comme ceci :

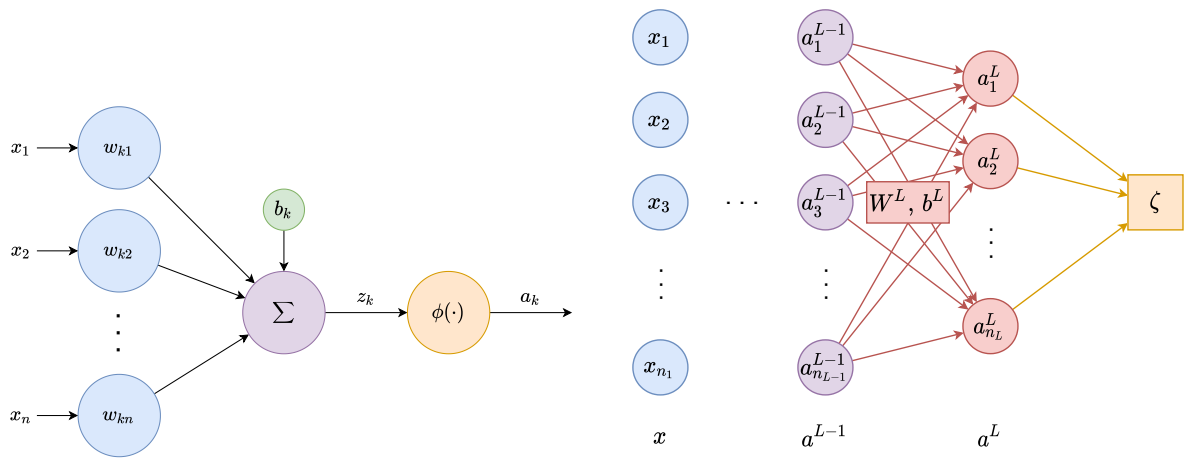
$$\begin{aligned} z_k &= \sum_{i=1}^{n} w_{ki} x_i \\ a_k &= \phi(z_k + b_k), \end{aligned} \tag{A.1}$$

où $(x_1, \cdots, x_n)$ sont des signaux d'entrée, a_k est le signal de sortie, $w_k = (w_{k1}, \cdots, w_{kn})$ sont les poids des synapses, b_k est appelé le biais et $\phi(\cdot)$ est une fonction différentiable non-linéaire appelée fonction d'activation. Tous les signaux et variables décrits sont des réels, $n \in \mathbb{N}$ est le nombre de synapses et $k \in \mathbb{N}$ est l'indice du neurone dans la couche du réseau.

Dans sa formulation la plus générique, un MLP est une fonction réelle décrite comme une composition des fonctions des couches successives de neurones. Sous forme vectorisée cela s'exprime :

$$\mathcal{F}(x) = \phi_L(W^L \cdots \phi_2(W^2 \phi_1(W^1 x + b^1) + b^2) \cdots + b^L), \tag{A.2}$$

où $L \in \mathbb{N}$ est le nombre de couches du MLP, W^l est la matrice des poids synaptiques de la couche l possédant n_l neurones (une ligne correspondant à un neurone) et b^l le vecteur



(a) Perceptron (b) MLP

FIGURE A.1 : (a) Schéma synthétique d'un perceptron, (b) schéma synthétique d'un réseau de neurones multicouches. Les notations mathématiques sont introduites dans le texte.

des biais de cette couche (les exposants représentent les indices de couches). Ce sont ces paramètres qui sont appris au cours de l'entraînement du réseau de neurones.

L'apprentissage est basé sur la minimisation de l'erreur du réseau, calculée par une fonction de coût ζ différentiable. La minimisation de cette erreur est réalisée via l'algorithme de la descente de gradient, qui permet de trouver un minimum local de $\mathcal{F}$ vis-à-vis de l'erreur. L'algorithme de la descente de gradient est basé sur le calcul du gradient de la fonction de coût par rapport aux paramètres à optimiser, c'est-à-dire les poids synaptiques et les biais. Cependant, le calcul analytique de cette dérivée n'est pas simple à réaliser dès que la taille du réseau n'est pas triviale et le coût calculatoire peut être élevé. Le calcul via les différences finies n'est pas envisageable non plus, le coût calculatoire explose rapidement au-delà de quelques dizaines de neurones. La solution à ce problème a été formalisée en 1985 dans [3] : il s'agit d'un algorithme efficace de calcul du gradient pour les graphes de calcul appelé *rétropropagation*. La complexité de cet algorithme est linéaire vis-à-vis du nombre de paramètres à optimiser et permet une utilisation efficace de la mémoire car le calcul est défini par récurrence grâce à une utilisation astucieuse de la règle de la chaîne de la dérivée.

L'algorithme de rétropropagation vise à minimiser l'erreur $\zeta(y, \mathcal{F}(x))$ où y est la vérité de terrain (le résultat attendu) de la donnée d'entrée x. Pour cela, il faut estimer le gradient $\nabla_{W^l}\zeta$ et $\nabla_{b^l}\zeta$ de chaque couche l du réseau. La règle de la chaîne permet d'écrire pour toute couche l :

$$\begin{aligned}\nabla_{W^l}\zeta &= \frac{d\zeta}{dW^l} \\ &= \frac{d\zeta}{da^L} \cdot \frac{da^L}{dz^L} \cdot \frac{dz^L}{da^{L-1}} \cdots \frac{da^{l+1}}{dz^{l+1}} \cdot \frac{dz^{l+1}}{da^l} \cdot \frac{da^l}{dz^l} \cdot \frac{dz^l}{dW^l}.\end{aligned} \tag{A.3}$$

Cette expression peut être réécrite en remplaçant les dérivées par l'expression de la dérivée évaluée depuis la forme vectorisée de l'équation A.1 :

$$\nabla_{W^l}\zeta = \frac{d\zeta}{da^L} \cdot \phi'_L \cdot W^L \cdots \phi'_{l+1} \cdot W^{l+1} \cdot \phi'_l \cdot a^{l-1}. \tag{A.4}$$

Ensuite on peut extraire le terme suivant qui désigne l'erreur du réseau à la couche l :

$$\delta^l = \frac{d\zeta}{da^L} \cdot \phi'_L \cdot W^L \cdots \phi'_{l+1} \cdot W^{l+1} \cdot \phi'_l, \tag{A.5}$$

et remarquer qu'il peut s'exprimer récursivement en fonction de l'erreur de la couche suivante, avec comme condition initiale l'erreur δ^L qui est connue :

$$\begin{aligned} \delta^l &= \delta^{l+1} \cdot W^{l+1} \cdot \phi'_l, \\ \delta^L &= \frac{d\zeta}{da^L}(a^L). \end{aligned} \tag{A.6}$$

Ceci permet d'exprimer le gradient de la fonction de coût vis-à-vis des poids synaptiques de la couche l de manière récursive :

$$\nabla_{W^l}\zeta = \delta^l \cdot a^{l-1}. \tag{A.7}$$

Cette représentation est efficace car elle permet de calculer l'erreur du réseau en commençant par la dernière couche, puis de "remonter" le réseau en arrière en calculant à chaque fois l'erreur et la valeur du gradient à chaque neurone. Cette représentation suppose que les activations a^l des neurones ont été mises en cache lors de l'évaluation de $\mathcal{F}(x)$. Un calcul similaire permet de calculer le gradient vis-à-vis des biais $\nabla_{b^l}\zeta$.

Ce gradient peut ensuite être utilisé afin de calculer la correction à apporter aux poids synaptiques et biais à chaque itération de la descente de gradient. Une itération de cet algorithme consiste en l'évaluation du gradient et le moyennage de celui-ci sur l'ensemble des échantillons d'apprentissage. Cela donne, pour un set de données d'entrée $\mathcal{X}$ et de vérités de terrain $\mathcal{Y}$ tous deux de cardinal N :

$$\theta_{t+1} = \theta_t - \frac{\eta}{N}\sum_{i=1}^{N} \nabla_\theta \zeta(\mathcal{Y}_i, \mathcal{F}(\mathcal{X}_i)) \tag{A.8}$$

où θ_t désigne la concaténation des paramètres d'apprentissage à l'itération t et η est un réel strictement positif appelé taux d'apprentissage (Learning Rate (LR)).

La descente de gradient n'est pas la seule méthode d'apprentissage des réseaux de neurones, d'autres méthodes ont vu le jour comme la descente de gradient stochastique (SGD) [16] et ADAM [70].

Il a été montré [58] que le MLP satisfait le théorème d'approximation universelle, il est donc capable d'approximer n'importe quelle fonction continue définie sur des sous-

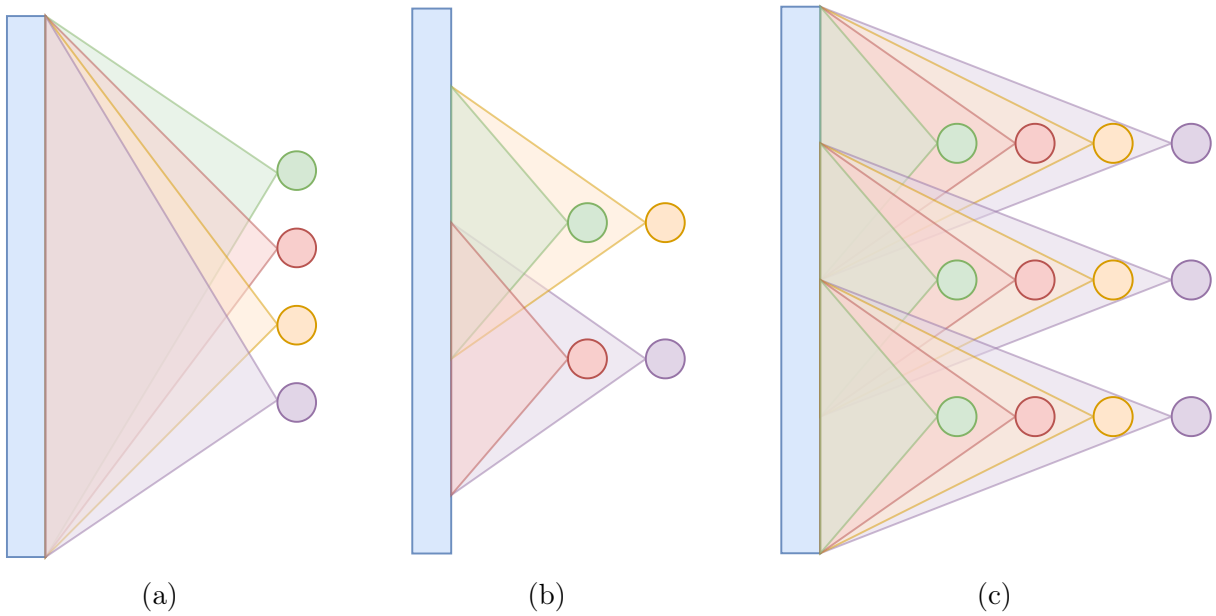


FIGURE A.2 : Schéma du principe de fonction des CNN illustré sur une couche de neurones. Le bloc bleu représente une donnée cohérente localement (une séquence temporelle d'échantillons, la couche de neurones précédente, etc.) et les points de couleurs représentent des neurones d'une couche. Les connections (champ de vision) sont matérialisées par les cônes de couleur. a représente un MLP (*Fully Connected* (FC)), b représente un réseau ayant des neurones connectés localement uniquement (champ de vision réduit) et c représente un CNN où les poids des neurones sont partagés pour toutes les positions (neurones de la même couleur).

ensembles compacts de R^n, avec n en entier positif. En revanche, cela ne garantit pas la convergence de la descente de gradient qui peut diverger selon les conditions initiales des poids synaptiques et des données d'entraînement.

A.2 Réseaux de neurones convolutifs

L'ensemble des réseaux de neurones convolutifs (*Convolutional Neural Networks* (CNN)) est un sous-ensemble des MLP couramment employé sur des images ou des séquences temporelles (courbes de température par exemple). Ils sont conçus pour réduire radicalement la complexité du réseau de neurones en tirant parti du partage de poids synaptiques et de la cohérence locale des données. En effet, dans un MLP chaque neurone d'une couche est connecté à tous les neurones de la couche précédente. D'abord, plusieurs neurones d'une même couche pouvant apprendre la même "fonction", cela rend cette approche très sensible au sur-apprentissage. Par ailleurs, l'aspect local de la donnée n'est pas pris en compte et chaque neurone apprend une "fonction" non localisée puisqu'il est connecté à tous les neurones de la couche précédente. Ce mode de fonctionnement est coûteux en temps de calcul et en empreinte mémoire, et peut générer du sur-apprentissage ou un apprentissage lent.

Pour palier à ce problème, les CNN sont construits autour de deux concepts illustrés dans la figure A.2 :

- La localité de l'activation neuronale. Chaque neurone n'est connecté qu'à un ensemble réduit de neurones "voisins" de la couche précédente. Le voisinage est défini soit par une proximité spatiale (un bloc de pixels dans une image par exemple), soit par une proximité temporelle (échantillons de données autour d'une certaine date par exemple). Ainsi, chaque neurone apprend une fonction locale plutôt que globale. Cette connexion locale est illustrée par la sous-figure b : les neurones ne sont pas connectés à tous les neurones de la couche précédente (ou de la donnée d'entrée) mais uniquement à un certain voisinage, contrairement au MLP (illustré dans la sous-figure a). Le neurone vert est responsable de la détection dans la partie haute de la donnée tandis que le neurone rouge est responsable de la détection dans la partie basse.

- Le partage des poids synaptiques. Les données comportant une cohérence locale, comme les images et les séquences temporelles, sont équivariantes par translation. Par exemple, un pic de température dans une courbe de température est une propriété qui ne dépend pas de sa position dans la séquence mais uniquement de l'aspect local de la courbe. L'apprentissage d'un neurone à une position donnée est donc pertinent à toutes les autres positions. Les poids synaptiques des neurones sont donc partagés pour toutes les positions. Sur la sous-figure c, il y a trois zones illustrées et chaque neurone parmi les quatre illustrés (le vert par exemple) est partagé sur toutes les zones.

L'avantage de ce fonctionnement est qu'il peut être implémenté très efficacement par des convolutions pouvant être facilement parallélisées sur du matériel spécialisé comme un GPU.

Les CNN tirent donc parti d'un a priori sur la cohérence de la structure des données pour limiter radicalement le nombre de neurones et d'opérations à réaliser tout en maintenant une excellente performance [82].

A.3 Principales couches des réseaux de neurones convolutifs

Le principal composant des CNN est la couche convolutive qui sont des neurones regroupés au sein de banques de *filtres*. Chaque filtre est composé de neurones ayant une activation locale et partagés pour l'ensemble des positions de la donnée d'entrée. Un CNN est composé de plusieurs couches convolutives successives extrayant une information sémantique de plus en plus abstraite et complexe formant une hiérarchie : les premières couches extraient des

motifs simples (couleurs, textures) tandis que les dernières couches extraient des concepts de haut niveau (formes complexes, objets).

Trois autres types de couches sont employés en plus des couches convolutives :

- Les non-linéarités. Ce sont des fonctions non-linéaires qui sont appliquées aux valeurs d'activation générées par les couches convolutives. Il existe une grande variété de fonctions non-linéaires couramment employées : la sigmoïde, la tangente hyperbolique, l'unité linéaire rectifiée [4] (*Rectified Linear Unit*, (ReLU)), etc. Utiliser une fonction non-linéaire permet d'apprendre des comportements autres que linéaires ce qui est le cas de la majorité des problèmes complexes.

- Les couches de normalisation. La normalisation de batch [62] (*BatchNorm* (BN)) est de loin la plus employée. Les couches de normalisation permettent d'améliorer significativement la vitesse de convergence des réseaux de neurones profonds mais la raison exacte du gain de performance est mal connue [118].

- Les couches de réduction de la dimension spatiale. Elles consistent en un sous-échantillonnage des cartes de descripteurs (les activations produites par les couches convolutives) afin d'augmenter le *champ de vision* du réseau et de réduire la complexité calculatoire (une définition des descripteurs plus faible implique un nombre d'opérations exécutées plus faible). Il s'agit d'un compromis : les cartes de descripteurs sont réduites donc la définition spatiale est moindre mais en contrepartie, chaque neurone des couches suivantes a un champ de vision démultiplié. Les fonctions de sous-échantillonnage les plus courantes sont la fonction maximum (*maxpool*) et la fonction moyenne (*averagepool*).

A.4 Apprentissage profond dans la pratique

Le principe de fonctionnement des MLP vu dans la section précédente n'est pas restrictif et ne se limite pas aux réseaux de neurones à couche FC. La rétropropagation fait partie d'un ensemble de techniques d'évaluation de gradient appelé Différentiation Automatique (AD). L'AD permet de calculer la dérivée de n'importe quel graphe de calcul défini programmatiquement à un ordre arbitraire, même si le graphe de calcul est cyclique et même s'il comporte des fonctions non différentiables [83]. Cela rend l'AD extrêmement générique et flexible, et surtout, permet d'abstraire toute la complexité du calcul de la descente de gradient lors de la programmation tout en garantissant de bonnes performances.

Cette capacité permet le développement rapide de réseaux de neurones à partir de primitives de calcul préexistantes qui peuvent être composées afin d'obtenir un réseau plus complexe. Le travail de développement est axé, d'une part, sur le design du réseau en lui-même et, d'autre part, sur la procédure d'entraînement. Le Deep Learning, contrairement aux méthodes de traitement traditionnelles, est une approche principalement *data-driven*.

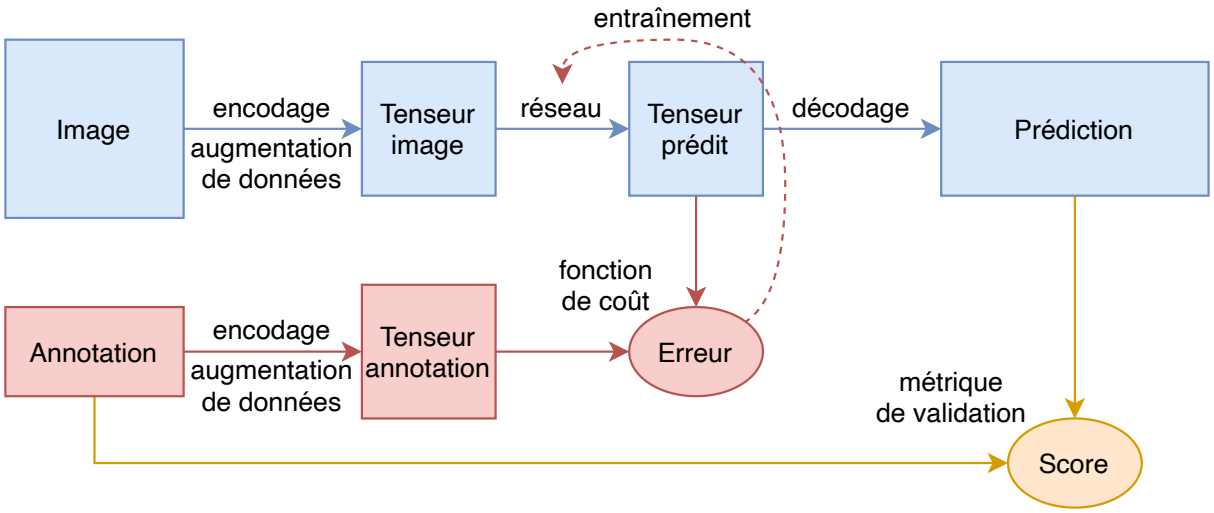


FIGURE A.3 : Schéma de la procédure d'entraînement d'un réseau de neurone profond de traitement d'image. Le fslux de données propre à l'inférence est signalé en bleu, le flux des données propre à l'apprentissage en rouge et celui de la validation en jaune.

Ce détail a son importance : la conception d'un algorithme de Deep Learning et son bon fonctionnement tiennent plus de la bonne qualité et quantité de données d'entraînement ainsi que du processus de transformation et de conversion du flux de données que de l'architecture du réseau[1]. Un schéma de la procédure générique de l'entraînement d'un réseau de traitement d'image (détection d'objet, classification, etc.) est présenté en figure A.3. Ce schéma met en valeur la chaîne de traitement des données et l'importance des étapes d'encodage et de décodage des données, i.e. la conversion de données brutes (images) et d'annotations humaines (listes d'objets, squelette de pose, etc.) en une représentation numérique (le tenseur) adaptée au réseau de neurones et aux contraintes numériques, et vice-versa. La fonction de coût a aussi une importance centrale puisque c'est elle qui supervise l'ensemble de l'apprentissage.

[1]Le développement de SDNet présenté en section 2.3 nécessite 83 lignes de Python pour la définition du réseau, contre plus de 1000 pour l'encodage, le décodage, le traitement et la visualisation des données.

Annexe B

Complément sur l'apprentissage de l'espèce des feuilles par SDNet

Il est possible de traiter l'organe "feuille" de deux manières :

- La première manière considère la feuille comme étant *agnostique* de l'espèce de la plante considérée. Dans ce cas, il n'y a qu'une seule classe "feuille" commune à toutes les espèces de plantes.
- La seconde manière considère la feuille comme étant *non-agnostique* de l'espèce. Dans ce cas, il y a une classe feuille par espèce de plante (par exemple "feuille de maïs", "feuille de haricot", etc.).

Pour les applications visées par le travail présenté dans ce manuscrit, il y a peu d'intérêt à envisager la manière non-agnostique. D'abord, la feuille n'est pas un organe utilisé pour le désherbage de précision visé par BIPBIP. Ensuite, l'estimation du nombre de feuilles des plantes ne requiert pas de distinguer l'espèce des feuilles mais seulement leur nombre. Pour finir, SDNet a l'avantage d'encoder implicitement la classe de feuille, même lorsque la manière agnostique est employée : l'espèce de la feuille correspond à l'espèce de la tige à laquelle elle est associée.

Néanmoins, il peut être intéressant d'évaluer les performances dans le cas non-agnostique afin de vérifier l'impact de ce choix. On pourrait, par exemple, faire l'hypothèse qu'un réseau de neurones profond apprenant explicitement l'espèce de chaque feuille pourrait permettre d'éliminer des faux positifs dus à des confusions d'espèces et serait donc plus performant. Pour vérifier cette hypothèse, une expérimentation rapide est menée sur le détecteur de structure SDNet. Il est entraîné sur la base de données DB_2 en prenant en compte deux ancres (maïs et haricot) et deux parties (feuille de maïs et feuille de haricot) plutôt qu'une seule partie agnostique de l'espèce de la plante. Le réseau est entraîné dans des conditions optimales déduites dans la section 3.3.2 : la définition est de 512×512, le noyau Gaussien a une taille de 10 % du côté le plus petit de l'image et le seuil de confiance du détecteur est de 40 %.

Feuille	Label	Rappel	Précision	Score F1	MAE_{loc}
Agnostique	Haricot	85,7 %	92,3 %	88,8 %	5,1 mm
	Maïs	93,1 %	96,0 %	94,5 %	3,7 mm
	Feuille	86,1 %	91,8 %	88,8 %	2,8 mm
	Total	87,0 %	92,4 %	89,6 %	3,3 mm
Non-agnostique	Haricot	86,5 %	92,3 %	89,3 %	5,6 mm
	Maïs	91,2 %	96,9 %	93,9 %	4,3 mm
	Feuille haricot	80,8 %	91,2 %	85,7 %	3,3 mm
	Feuille maïs	82,6 %	92,3 %	87,2 %	3,2 mm
	Total	83,8 %	92,6 %	88,0 %	3,8 mm

Tableau B.1 : Comparaison des performances de SDNet sur la tâche de localisation de points d'intérêt selon que les feuilles sont apprises de manière agnostique ou non de l'espèce de la plante. La base de données utilisée est DB_2, la définition d'entrée est de 512, le noyau Gaussien a une taille de 10 % et le seuil de confiance est de 40 %.

Le tableau B.1 compare les performances de SDNet évaluées sur la tâche de localisation de points d'intérêt selon que la classe feuille soit apprise de manière agnostique ou non de l'espèce de la plante. Les erreurs standard ne sont pas présentées et sont similaires en ordre de grandeur à celles présentées dans la section 3.4.1.1, soit approximativement entre 0,5 % et 2,0 %. Globalement, le Score F1 (total) de l'approche non-agnostique est inférieur de 1,6 %. Cette chute est due à une baisse du Rappel de 3,2 % puisque la Précision est équivalente entre les deux (0,2 % de différence). Le score de localisation MAE_{loc} est supérieur de 0,5 mm pour l'approche non-agnostique, qui est donc légèrement moins précise en localisation. Concernant les résultats par classe de points d'intérêt, c'est principalement la qualité de la détection des feuilles qui impacte négativement les performances. Pour l'approche non-agnostique, le Rappel est inférieur de 5,3 % pour la feuille de haricot et inférieur de 3,5 % pour la feuille de maïs en comparaison du cas agnostique.

Cette expérimentation montre que SDNet est modérément plus performant lorsque l'espèce des feuilles n'est pas apprise explicitement. Il n'y a pas de cause évidente expliquant ce phénomène et la différence de performance est peu significative. La manière agnostique pour détecter les feuilles est choisie pour le travail présenté dans ce manuscrit puisque, d'une part, la classe des feuilles n'est pas requise pour les tâches évaluées et, d'autre part, SDNet est implicitement capable de prédire l'espèce des feuilles si besoin, sans entraînement spécifique.

Annexe C

Hypothèse de déplacement uniforme du module de vision BIPBIP

L'algorithme d'agrégation temporelle présenté dans la section 2.4 s'appuie sur plusieurs hypothèses sur la configuration de la caméra du bloc-outil BIPBIP. Notamment, une hypothèse faite est que la caméra étant face au sol (axe optique perpendiculaire au plan du sol) et son déplacement étant un mouvement de translation, le déplacement du sol perçu est identique en toute position de l'image. Sous cette hypothèse, le flux optique (déplacement en pixels) est identique en tous points de l'image et correspond, à un coefficient multiplicateur près, au déplacement de la caméra exprimé dans le référentiel lié au sol.

Cette annexe présente une démonstration de cette hypothèse de déplacement uniforme. Les hypothèses faites sont listées ci-après :

- La caméra est supposée être étalonnée, sans distorsion optique et est assimilée à une caméra *pinhole*.
- Le sol est parfaitement plan.
- L'axe optique (OG) (droite en rouge sur la figure) est perpendiculaire au sol.
- Le déplacement de la caméra exprimé dans le repère lié au sol est rectiligne.
- L'altitude de la caméra h est fixe (distance du foyer F au sol).

La figure C.1 présente un schéma simplifié du problème. Elle présente une coupe de côté du capteur de la caméra (plan image) et du sol. D'abord, l'objectif est de montrer que chaque pixel représente une aire identique, quelle que soit sa position sur le capteur. Le problème étant symétrique, la démonstration est réalisée pour un seul des deux axes du capteur. Le problème se résume donc à montrer que chaque pixel de largeur a représente la même longueur y au sol, i.e. la longueur représentée par un pixel est indépendante de sa position x sur le capteur (cf. schéma).

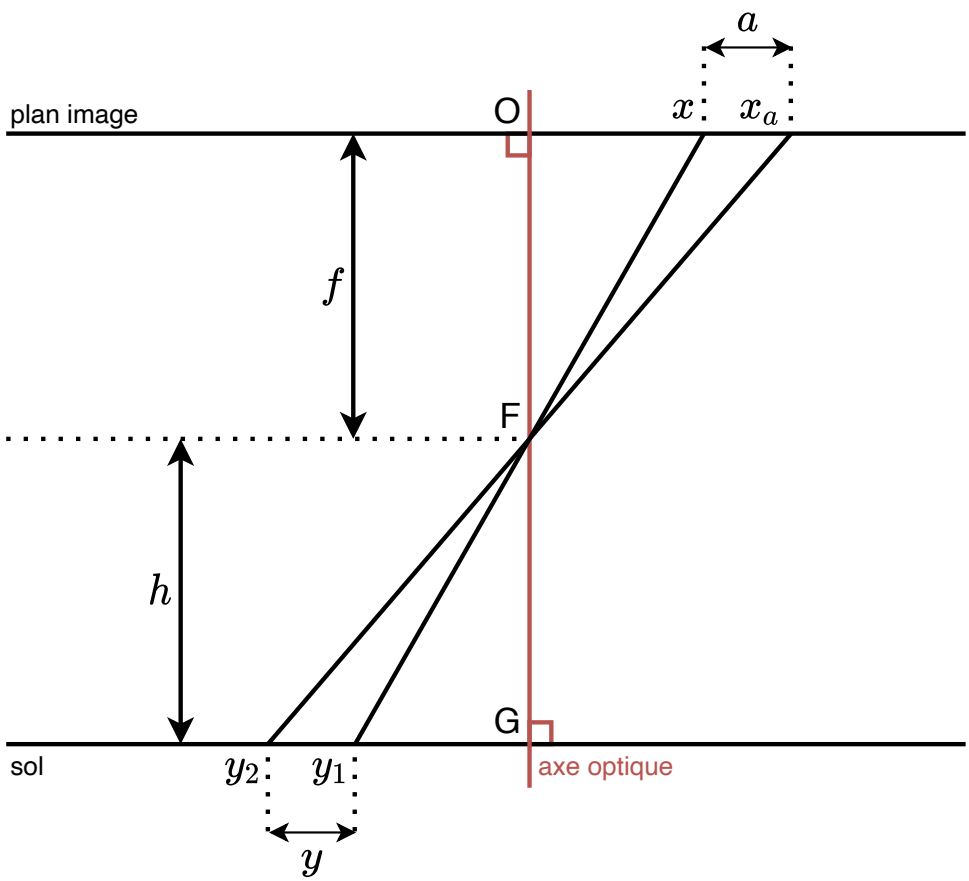


FIGURE C.1 : Coupe schématique de la caméra du bloc-outil BIPBIP dans le plan formé par l'axe optique (en rouge) de la caméra et de l'un des axes du capteur (plan image (Ox)). f est la distance focale, h l'altitude de la caméra par rapport au sol et a la taille d'un pixel p situé à une distance variable $\overline{Ox}$ du centre du plan image.

On note f la distance focale de la caméra, h la distance du foyer F au point du sol G le plus proche et a la largeur d'un pixel p situé à une distance variable x du centre du plan image O. Sous les conditions énoncées plus haut, les rayons optiques (xy_1) et $(x_a y_2)$ partants des extrémités du pixel p sont des droites et le plan image (Ox) est parallèle au sol (Gy_1). On peut donc appliquer le théorème de Thalès trois fois. Une première fois sur les triangles OFx et FGy_1 :

$$\frac{\overline{Ox}}{\overline{Gy_1}} = \frac{\overline{Fx}}{\overline{Fy_1}}, \tag{C.1}$$

et une deuxième fois sur les triangles xx_aF et Fy_1y_2 :

$$\frac{a}{y} = \frac{\overline{Fx}}{\overline{Fy_1}}. \tag{C.2}$$

Par égalité, on obtient :

$$\frac{a}{y} = \frac{\overline{Ox}}{\overline{Gy_1}}. \tag{C.3}$$

On peut également appliquer une troisième fois le théorème de Thalès aux triangles FOx et FGy_1 :

$$\frac{\overline{Ox}}{\overline{Gy_1}} = \frac{f}{h}. \tag{C.4}$$

Finalement, pas substitution dans l'équation C.3 :

$$y = \frac{af}{h} \tag{C.5}$$

Dette dernière égalité montre que la longueur au sol y représentée par un pixel p de taille a est indépendante de la position x du pixel sur le capteur. Elle ne dépend que de la taille du pixel a, de la distance focale f et de l'altitude h du capteur qui sont tous les trois des paramètres fixes. Dans ces conditions, une translation du capteur sur l'axe (Ox) se traduit nécessairement par un déplacement du sol identique en tous points du capteur.

Annexe D

Illustrations des bases de données et des prédictions

La figure D.1 présente des échantillons d'images tirés aléatoirement dans la base de données DB_1.

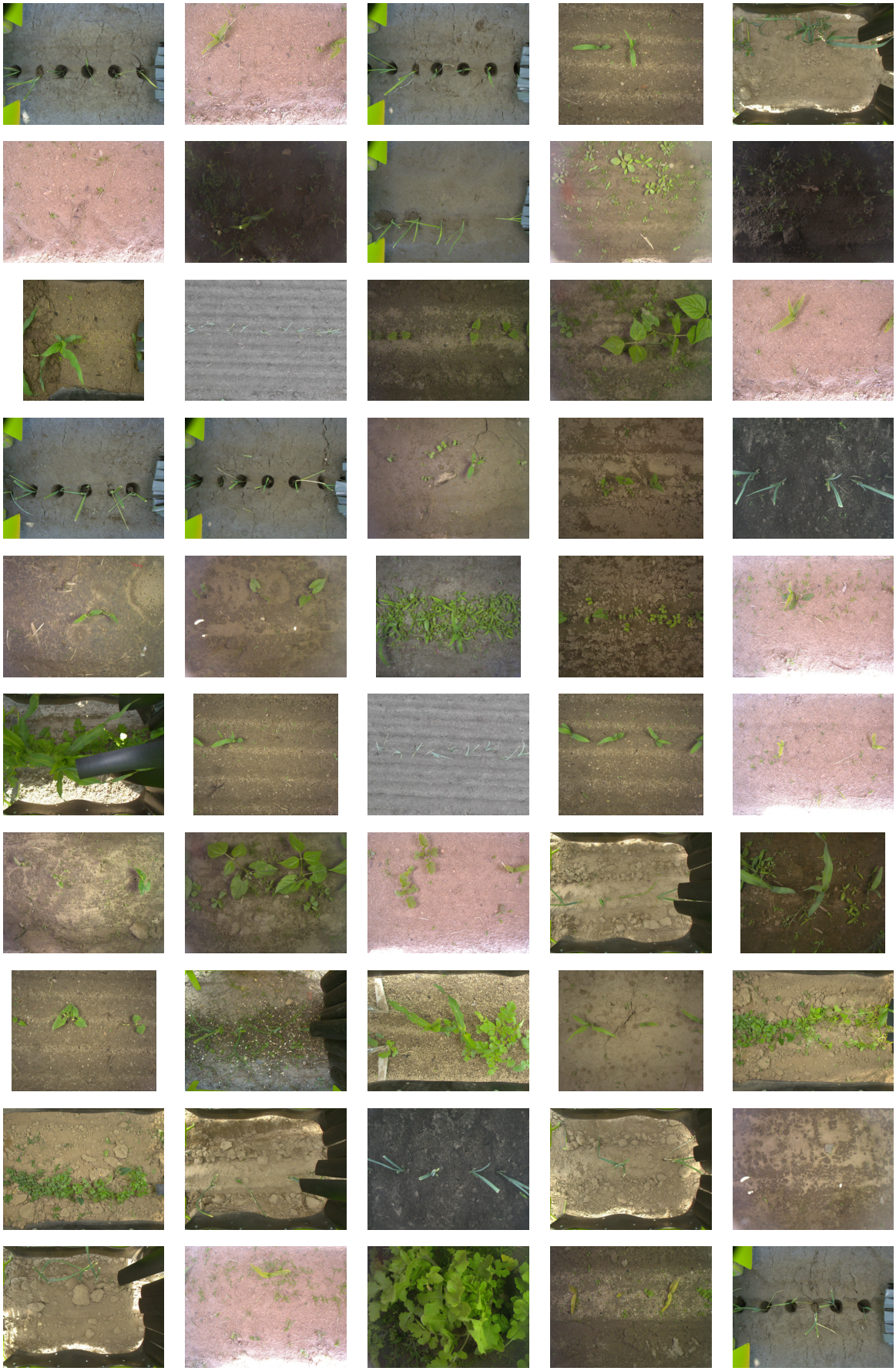


FIGURE D.1 : Échantillons d'images de la base de données DB_1 tirés aléatoirement.

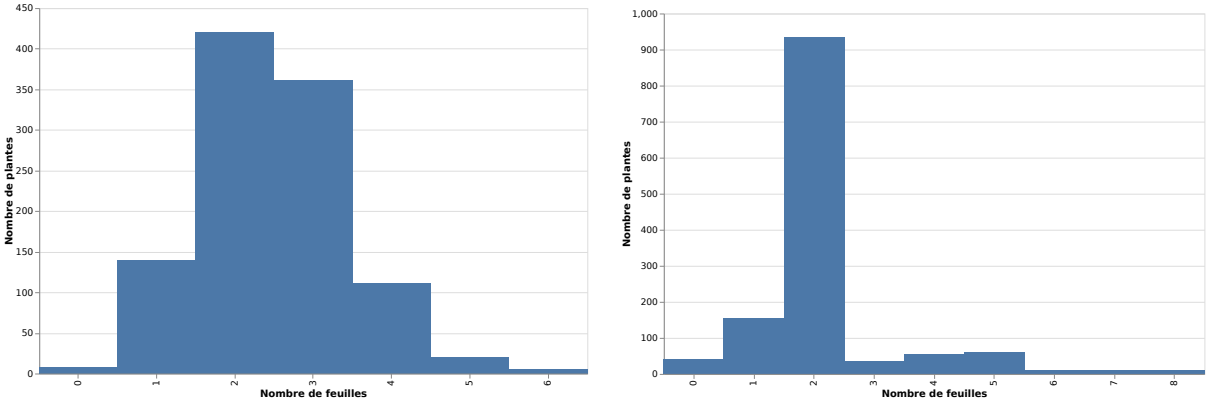
Annexe E

Distribution du nombre de feuilles de la base de données DB_2

La distribution du nombre de feuilles par plante est inégalement répartie dans la base de données de structure de plante DB_2. D'une part, comme présenté dans la section 1.1 le stade de développement précoce est visé, donc les plantes sont encore petites et ont peu de feuilles. La figure E.1 présente l'histogramme du nombre de feuilles par plante pour les deux variétés de plantes présentes dans la base de données DB_2.

Cette figure montre que le nombre moyen de feuilles par plante se situe autour de 2,5 pour le maïs et autour de 2 pour le haricot. La distribution est plus étalée pour le maïs et est centrée autour d'un seul mode situé à la moyenne. La distribution du haricot possède deux modes, un dominant centré sur la moyenne et l'autre autour de 4,5 feuilles. Ce second mode est causé par quelques images de haricot prises à un stade plus tardif où des tiges et feuilles secondaires ont pu se développer.

Ces histogrammes montrent que la distribution du nombre de feuilles par plante est inégale et que la base de données DB_2 ne possède que peu d'échantillons de plantes à



(a) Maïs (b) Haricot

FIGURE E.1 : Histogrammes du nombre de feuilles par plante pour la base de données DB_2, ventilée par variété de plante.

un stade de développement plus avancé. Ce phénomène n'est pas problématique puisque l'application de désherbage visée se concentre sur le stade précoce, néanmoins les résultats présentés, notamment sur l'estimation du nombre de feuilles, sont à mettre en contraste avec ces observations.

Annexe F

Considérations sur le risque de sur-apprentissage et sur la capacité de généralisation des réseaux entraînés

Les réseaux de neurones sont sujets au phénomène de sur-apprentissage (*overfitting*) pouvant mener à des performances dégradées sur de nouvelles données n'ayant pas servi pour l'apprentissage. Le sur-apprentissage désigne la capacité du réseau de neurones à apprendre "par cœur" les données d'entraînement, ce qui se fait au détriment de sa capacité de généralisation. Un réseau qui a sur-appris sera donc performant lorsqu'il est évalué sur ses données d'entraînement mais médiocre sur des données qu'il n'a jamais observées. Afin de réduire le risque de sur-apprentissage lors de l'entraînement du réseau, plusieurs techniques peuvent être mises en place :

- L'utilisation d'une grande quantité de données d'entraînement. Une grande quantité de données empêche le réseau de mémoriser tous les cas de figure puisque celui-ci a une capacité limitée. Une plus grande quantité de données permet aussi de couvrir une plus grande diversité de données, obligeant le réseau à apprendre des paramètres moins spécifiques.

- L'utilisation d'augmentations de données. Les augmentations de données permettent de créer de nouvelles données d'entraînement à partir de celles existantes tout en conservant la sémantique de la donnée. Dans le domaine des CNN opérant sur des images, des augmentations courantes sont les changements de couleur (saturation, luminosité, contraste), l'ajout de bruit (floutage) et les modifications géométriques de l'image (miroir horizontal, changement de résolution, rotation, etc.) [107]. Cette technique permet d'augmenter artificiellement la quantité de données d'entraînement et a donc le même effet qu'une augmentation du nombre de données.

- L'utilisation de techniques de régularisation. Ces techniques ont pour but d'empêcher l'entraînement du réseau d'atteindre un minimum local de la fonction de coût.

Conceptuellement, les techniques de régularisation permettent de "lisser" la fonction de coût afin de guider l'apprentissage vers un optimum plus global, et donc plus général. Parmi ces techniques il y a la normalisation de batch (*Batch norm*, (BN)) [62], le *dropout* [57], le déclin des poids du réseau (*weight decay*) [50] ou d'autres techniques plus avancées de régularisation de la fonction de coût [93].

En parallèle de ces techniques, il est également possible de détecter le sur-apprentissage en utilisant un second jeu de données nommé jeu de validation et contenant des données qui ne sont pas utilisées lors de l'apprentissage. Calculer les performances sur ce jeu de données isolé permet d'évaluer de manière non biaisée la capacité du réseau de neurones à généraliser sur de nouvelles données. Le sur-apprentissage est alors mis en évidence de deux manières :

- La courbe des performances du réseau au cours du temps lors de l'entraînement calculée sur le jeu de validation présente une régression de la métrique après une période initiale d'augmentation alors que la performance continue de s'améliorer sur le jeu d'entraînement. Cette désynchronisation de l'augmentation des performances montre que le réseau se spécialise trop sur les données du jeu d'entraînement, l'empêchant de généraliser et de progresser sur le jeu de validation.

- La performance calculée sur le jeu de validation est significativement plus basse que celle calculée sur le jeu d'entraînement. Ce comportement met en évidence une incapacité du réseau à généraliser à de nouvelles images.

Il existe un autre biais d'apprentissage différent du sur-apprentissage mais se matérialisant aussi par une régression de la capacité de généralisation. Les hyperparamètres du réseau de neurones sont généralement réglés (*model tuning*) sur le jeu de validation dans le but d'optimiser davantage les performances. Afin de ne pas introduire de biais de sélection sur la valeur optimale des hyperparamètres, un troisième jeu de données, nommé jeu de test, est idéalement employé pour évaluer la performance réelle du réseau. Les données de ce jeu de test ne sont ni utilisées lors de l'entraînement, ni utilisées lors de la recherche des hyperparamètres optimaux. Bien qu'idéalement requis, ce jeu de données est en pratique moins essentiel que le jeu de validation, le biais sur la valeur des hyperparamètres introduite par son absence étant moins important que le biais de sur-apprentissage.

En effet, les réseaux de neurones ont théoriquement la capacité d'approximer n'importe quelle fonction de façon aussi précise que nécessaire [58]. Échouer à obtenir une performance correcte avec un réseau de neurones ne peut provenir que (i) d'un entraînement non adapté, (ii) d'un nombre insuffisant de neurones ou (iii) d'un nombre trop faible et d'une qualité trop pauvre de données d'entraînement [58]. Un réseau qui sur-apprend a donc la capacité d'apprendre une fonction très différente de celle ciblée par l'apprentissage. En revanche, les hyperparamètres d'un réseau n'ont pas cette capacité d'apprentissage. Ils sont aussi bien

moins nombreux[1] que le nombre de paramètres neuronaux et influencent la performance à un niveau plus général. Dans ces conditions il est statistiquement peu probable qu'une certaine configuration des hyperparamètres favorise radicalement un jeu de données plutôt qu'un autre représentant la même distribution statistique[2].

On peut donc considérer que le jeu de test est une précaution utile pour éviter certains biais de sélection des hyperparamètres. En revanche, sa présence n'est pas nécessaire pour montrer qu'un réseau est capable de généraliser et ne sur-apprend pas : c'est le rôle du jeu de validation. La section suivante présente des expérimentations founissant des éléments de réponse quant à la capacité de généralisation du réseau SDNet.

F.1 Expérimentations sur la capacité de généralisation de SDNet

Les expérimentations présentées dans le chapitre 3 n'emploient pas de jeu de données de test. Ce choix a été réalisé en estimant que le biais sur le choix des hyperparamètres serait faible. Les jeux de données utilisés dans ce travail sont petits (moins de 2500 images pour la base de données la plus grande, DB_1) et ne pas utiliser de jeu de test a permis d'avoir plus de données d'entraînement et de validation disponibles. Un jeu d'entraînement plus fourni permet de réduire le sur-apprentissage et un jeu de validation plus grand permet de présenter des résultats statistiquement plus significatifs.

Néanmoins, afin de garantir un certain niveau de généralisation des réseaux, les techniques courantes de réduction du sur-apprentissage ont été utilisées pour les entraînements (augmentation de données, régularisation, etc. Voir la section 3.2). La manifestation du phénomène de sur-apprentissage a aussi été surveillée dans les courbes d'apprentissage. Les courbes d'apprentissage présentées dans la section 3.2 ne mettent pas en évidence ce phénomène : les performances évaluées sur le jeu de validation ne montrent pas de chute de la valeur d'une métrique à partir d'un instant donné, la variation des scores est uniquement à la hausse.

Malgré les bons résultats obtenus, il est néanmoins intéressant de quantifier le sur-apprentissage, i.e. la différence de performance entre le jeu d'entraînement et celui de validation. De plus, afin d'évaluer le biais possible sur le choix de la valeur des hyperparamètres, un petit jeu de test a été annoté. Ce jeu de données est relativement petit (73 images) mais permet d'identifier qualitativement un potentiel biais. Cette expérimentation est présentée pour SDNet uniquement et dans sa configuration optimale (seuil de confiance de 40 %, seuil de distance du décodeur de 10 %, taille du noyau gaussien de l'encodeur de

[1] Les hyperparamètres sont moins nombreux de plusieurs ordres de grandeurs pour les réseaux profonds contenant des millions voir des milliards de paramètres entraînables.

[2] Les jeux de données d'entraînement, de validation et de test sont censés représenter la même distribution de probabilité, i.e. les données sont “comparables" entre les différents jeux de données et distribuées de la même manière.

Jeu de données	Maïs	Haricot	Feuille	Total
Entraînement	98,2 %	95,9 %	94,8 %	95,5 %
Validation	94,5 %	88,8 %	88,7 %	89,6 %
Test	91,9 %	93,2 %	89,8 %	90,7 %

TABLEAU F.1 : Score F1 obtenu par SDNet dans sa configuration optimale sur la tâche de localisation de points d'intérêt sur les trois jeux de données : entraînement, validation et test (base de données DB_2).

10 %, compensation de la décimation activée).

Le tableau F.1 présente le Score F1 sur la tâche de localisation de points d'intérêt sur les trois jeux de données (entraînement, validation, test) et ventilé par type de point d'intérêt (base de données DB_2). Il montre sans surprise que SDNet est plus performant sur le jeu d'entraînement que sur celui de validation. Tous types de points d'intérêt confondus, il obtient un Score F1 5,9 % plus élevée sur le jeu d'entraînement que sur celui de validation. Il y a donc un sur-apprentissage bien que son ampleur soit modérée, le Score F1 sur le jeu de validation étant élevé. La comparaison entre le Score F1 calculé sur le jeu de validation et celui calculé sur le jeu de test montre un avantage de 1,1 % pour le set de test. Cet écart faible tend à montrer que le choix des hyperparamètres n'induit pas de biais dans l'évaluation, même si la petite taille du jeu de test rend la conclusion peu significative.

Ces résultats laissent penser que le phénomène de sur-apprentissage est limité pour le réseau SDNet et que le biais sur le choix des hyperparamètres n'est pas significatif. SDNet semble faire preuve d'une capacité de généralisation correcte. Ces conclusions sont consolidées par les résultats des évaluations Operose présentés dans l'annexe G qui font état de bonnes performances (F-Mesure) sur des données jamais observées et présentant potentiellement un décalage de domaine (*domain shift*).

Annexe G

Résultats des évaluations par Operose

Dans le cadre du Challenge ROSE (cf. section 1.1), le fonctionnement des systèmes de désherbage précoce de précision est évalué plusieurs fois par an lors d'évaluations de terrain. Un plan d'évaluation, conçu par le consortium Operose composé du LNE[1] et de l'INRAE[2] de Montoldre, est mis en place afin de mesurer scientifiquement la capacité de désherbage des systèmes. Deux plantes d'intérêt sont considérées, le maïs et le haricot, ainsi que quatre types d'adventices : la moutarde, la matricaire, le ray-grass et le chénopode. Le schéma de semis des plantes d'intérêt est présenté dans la section 1.1. La densité de semis des adventices est de 27 graines par mètre linéaire.

L'une des tâches évaluées par Operose est la détection et la localisation des plantes d'intérêt et des adventices. Deux planches de cultures dédiées à cette tâche, l'une sur le maïs et l'autre sur le haricot, sont préparées par Operose. Les systèmes de désherbage de précision y réalisent ensuite des acquisitions de données dans des conditions similaires aux conditions réelles de désherbage. Les prédictions de détection sont ensuite calculées a posteriori puis envoyées au consortium Operose pour l'évaluation. Cette annexe présente les résultats obtenus par BIPBIP lors des deux premières évaluations en 2019 et en 2020[3] sur les deux plantes d'intérêt évaluées (BIPBIP ne détecte pas les adventices).

La méthode et les métriques d'évaluation proposées par Operose sont différentes de celles présentées dans ce manuscrit afin de convenir aux autres solutions technologiques. Ainsi, elles sont basées sur des masques de segmentation des plantes. Le fonctionnement des métriques est néanmoins similaire au processus en quatre étapes présenté dans la section 3.1.2 :

- La métrique de similarité est basée sur l'aire de l'union disjointe entre deux instances. Plus cette aire est faible, plus les instances sont similaires. Deux instances ayant une intersection nulle sont dissimilaires et deux instances ayant une union disjointe nulle sont parfaitement similaires. Dans le principe, cette mesure est semblable à

[1] Laboratoire National de Métrologie et d'Essais
[2] Institut National de Recherche pour l'Agriculture, l'Alimentation et l'Environnement
[3] Les résultats de l'évaluation de 2021 n'étaient pas connus au moment de la rédaction du document.

l'*Intersection over Union* (IoU).

- L'association optimale des prédictions aux vérités de terrain (étape nommée *mapping* par Operose) consiste en l'association minimisant la somme des aires des unions disjointes. À l'issue de ce processus, les prédictions non associées (soit parce qu'elles n'ont pas d'aire en commun avec une vérité de terrain, soit parce qu'elles sont déjà associées à d'autres vérités de terrain) sont des faux positifs (FP). Les masques de vérité de terrain non associés sont des faux négatifs (FN). Les prédictions associées avec une vérité de terrain d'une classe différente (prédiction de maïs sur du haricot, prédiction de haricot sur une adventice, etc.) sont des confusions (C). Les associations fructueuses sont quant à elles des vrais positifs (TP).

- Ces indicateurs (TP, FP, et FN) sont sommés sur l'ensemble des images de test.

- Les métriques quantifiant la pertinence de l'association sont le Rappel, la Précision et le Score F1 (cf. section 3.1.2) ainsi que l'EGER (*Estimated Global Error Rate*).

Il faut noter que même si les métriques d'Operose sont basées sur des masques de segmentation, la qualité des masques n'est pas évaluée. Les masques sont simplement utilisés lors de l'étape de *mapping* pour générer l'appariement optimal des vérités de terrain aux prédictions. Les métriques Operose évaluent donc bien une tâche de détection et de localisation et non pas une tâche de segmentation[4].

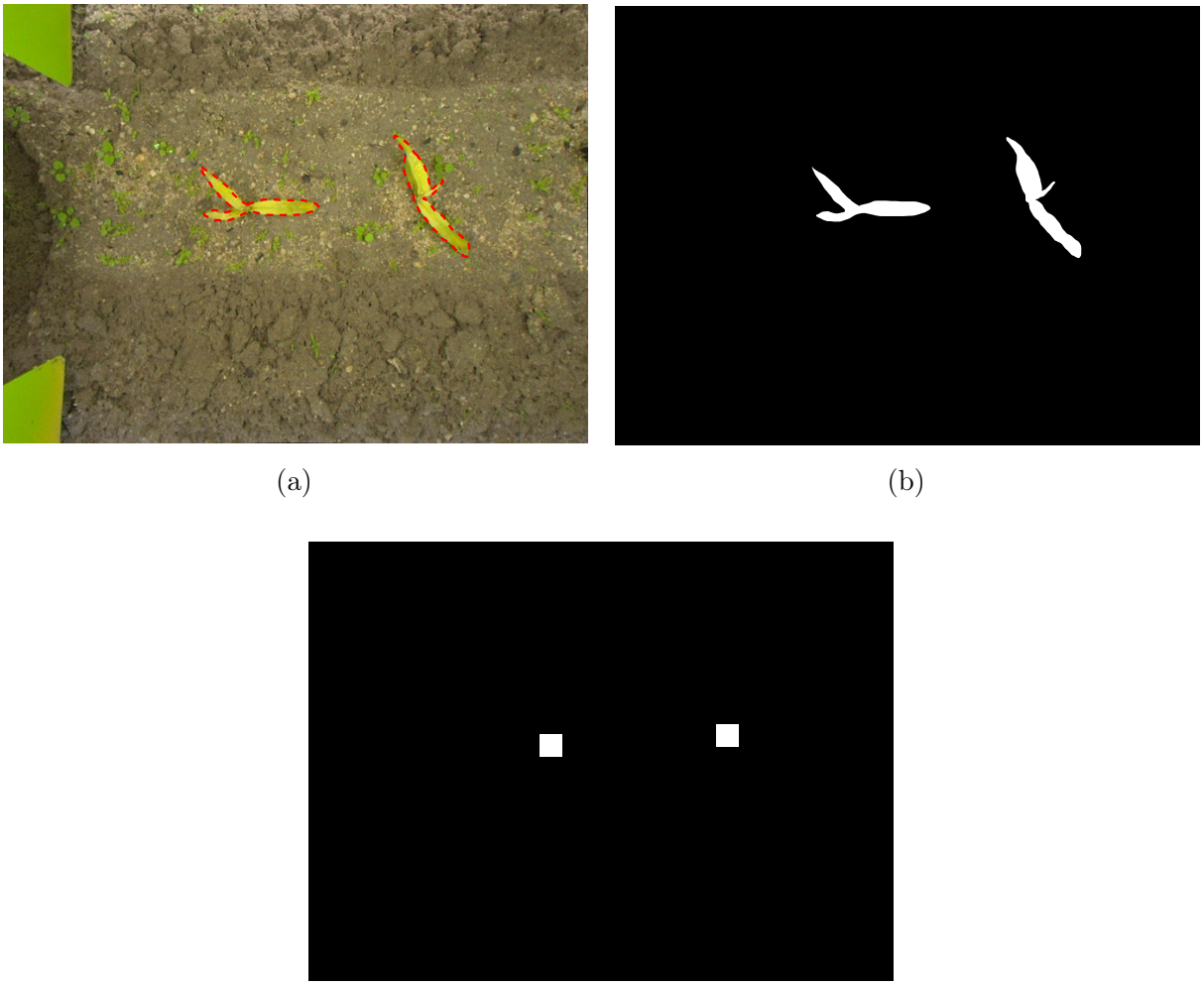
L'EGER est une métrique synthétisant les trois types d'erreurs : les oublis (FN), les confusions (C) et les fausses alarmes (FP). Il est défini comme :

$$\mathrm{EGER} = \frac{\sum_{k=1}^{N} k_2 \mathrm{C}_k + k_1 \mathrm{TP}_k + k_1 \mathrm{FN}_k}{\sum_{k=1}^{N} \mathrm{NR}_k}, \tag{G.1}$$

avec C_k, TP_k et FN_k respectivement la somme des confusions, la somme des fausses alarmes et la somme des oublis sur l'image k, N le nombre d'images et NR_k le nombre de prédictions réalisées sur l'image k. Les types d'erreurs ne sont pas tous équivalents. Ainsi, la pénalité pour un oubli ou une fausse alarme est de $k_1 = 1$ tandis que la pénalité pour une confusion est de $k_2 = 2$. L'EGER est exprimé en pourcentage. Il est de 0 % pour une détection parfaite et peut dépasser 100 % lorsqu'il y a trop d'erreurs de détection.

Le bloc-outil BIPBIP utilisant uniquement la position de la base des tiges pour réaliser le désherbage de précision, l'algorithme de détection utilisé (YOLOv4 Tiny pour les deux évaluations présentées dans cette annexe) ne prédit pas des masques de plante mais des points d'intérêts. Afin d'adapter l'algorithme de détection pour l'évaluation par Operose, les points d'intérêts doivent donc être convertis en masques. La conversion choisie consiste en la transformation de chaque point d'intérêt de coordonnée (x, y) en une boîte englobante

[4]La précision des masques de prédiction a évidemment une influence sur ces métriques. L'impact est cependant secondaire par rapport à la localisation du masque dans l'image.



(a) (b)

(c)

FIGURE G.1 : Illustration des masques de vérité de terrain et de prédiction utilisés pour les métriques d'Operose. (a) illustre une image captée par BIPBIP sur des plants de maïs (entourés en rouge), (b) présente le masque de vérité de terrain et (c) présente les masques prédits par BIPBIP. Pour l'évaluation des performances, il suffit que le masque prédit ait une intersection non nulle avec la vérité de terrain correspondante.

de centre (x, y) et de taille (w, h) où $w = h$ (boîte carrée). La taille de la boîte est fixée à une valeur de 5 % de la hauteur de l'image. La méthode de conversion choisie ainsi que ses paramètres n'ont pas fait l'objet d'expérimentations. La taille de la boîte englobante est simplement choisie pour être en cohérence avec les valeurs de taille de boîte et de seuil utilisés dans le manuscrit. Cette méthode de conversion est suffisante pour un usage avec les métriques Operose. En effet, le principal est que le masque de prédiction intersecte la vérité de terrain correspondante et n'intersecte pas les autres.

La figure G.1 illustre des masques de vérité de terrain et de prédiction sur une image captée par BIPBIP lors d'une évaluation de terrain. (G.1a) illustre une image avec les plantes de maïs entourées en rouge, (G.1b) présente les masques de vérités de terrain associés et (G.1c) illustre des exemples de masques prédits par BIPBIP.

La figure G.2 illustre les performances sur la tâche détection d'Operose pour les deux

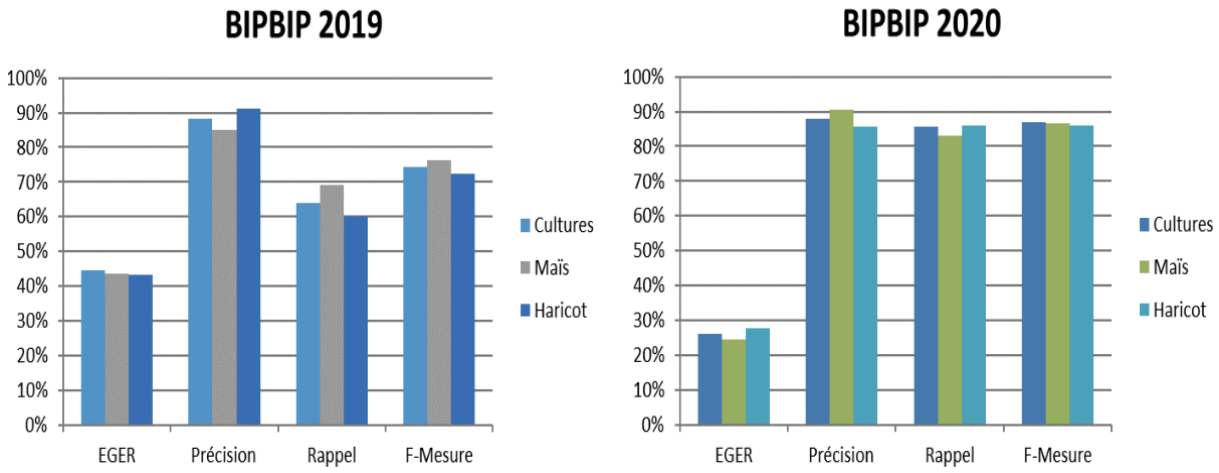


FIGURE G.2 : Résultats sur la tâche détection calculés par Operose pour les campagnes d'évaluation de 2019 (à gauche) et 2020 (à droite). YOLOv4 Tiny est utilisé comme détecteur de plantes d'intérêt.

dernières campagnes d'évaluation disponibles (2019 et 2020). Les résultats sont produits en utilisant YOLOv4 Tiny avec un seuil de confiance de 25 % et une taille de boîte englobante de tige de 7,5 % de la hauteur de l'image. Les bases de données d'entraînement ne sont en revanche pas les mêmes pour les deux campagnes puisque le réseau est entraîné chaque année sur des données plus récentes et plus complètes. Il faut aussi noter que malgré le contrôle des conditions expérimentales par Operose, les conditions de terrains peuvent varier d'une année à une autre. Les résultats des deux campagnes peuvent donc ne pas être comparables.

Les résultats de la dernière campagne (2020) montrent que le Score F1 est d'environ 85 % pour les deux types de cultures. Le Rappel est entre 80 % (maïs) et 85 % (haricot) tandis que la Précision est entre 85 % (haricot) et 90 % (maïs). Le score EGER est de 25 % tous types de plantes confondus. Les résultats de la campagne de 2019 montrent des résultats moins élevés : globalement, le Score F1 n'était que de 75 % et l'EGER était 20 % plus élevé. Le Rappel n'était pas équilibré avec la Précision. Cette dernière était de 90 % alors que le Rappel était de 65 %. Les meilleurs résultats de 2020, qu'ils soient causés par des conditions de terrain plus favorables ou par un meilleur algorithme, sont donc principalement dus à une moindre proportion d'oublis.

Les résultats de la dernière campagne (2020) sont cohérents et similaires en ordre de grandeur à ceux présentés dans la section 3.4.1.1. Néanmoins, ces résultats ne montrent pas de différence de performance significative selon le type de plante d'intérêt contrairement à l'étude présentée dans ce manuscrit.

Les méthodes développées ont été confrontées aux méthodes des trois autres consortiums en lice. Cependant, à ce jour, les résultats et les méthodes concurrentes n'ont pas été rendues publiques. Elles ne sont donc pas reproduites dans ce manuscrit.

Publications et actes de communication

Revues internationales à comité de lecture

- Crop stem detection and tracking for precision hoeing using deep learning. *Computers and Electronics in Agriculture (COMPAG)*, 2022.
- An annotated datatset of vegetable crops at an early stage of growth for proximal sensing applications. *Data in Brief*, 2022.

Conférences avec acte de communication publié

- SDNet : Unconstrained Object Structure Detector Network for In-Field Real-Time Crop Part Location and Phenotyping. *British Machine Vision Conference (BMVC)*, 2021. - Article et communication orale.
- BIPBIP : a mechanical and automated intra-row weeding solution. *International Horticultural Congress (IHC)*. Résumé accepté.
- Deep Learning for Early Weed vs. Crop Discimination Applied to Intra-Row Hoeing of Vegetables. *European Conference on Precision Agriculture (ECPA)*. 2019. - Poster et résumé.

Prix

- Le SIVAL d'argent a été attribué au bloc-outil de désherbage précoce BIPBIP en 2021.

Bibliographie

[1] Florent Abdelghafour, Barna Keresztes, Christian Germain et Jean-Pierre da Costa : In field detection of downy mildew symptoms with proximal colour imaging. *Sensors*, 20(16):4380, août 2020.

[2] Florent Abdelghafour, Roxana Rosu, Barna Keresztes, Christian Germain et Jean-Pierre da Costa : A Bayesian framework for joint structure and colour based pixel-wise classification of grapevine proximal images. *Computers and Electronics in Agriculture*, 158:345, mars 2019.

[3] David H. Ackley, Geoffrey E. Hinton et Terrence J. Sejnowski : A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, janvier 1985.

[4] Abien Fred Agarap : Deep Learning using Rectified Linear Units (ReLU). mars 2018.

[5] Sara Akodad, Lionel Bombrun, Junshi Xia, Yannick Berthoumieu et Christian Germain : Ensemble Learning Approaches Based on Covariance Pooling of CNN Features for High Resolution Remote Sensing Scene Classification. *Remote Sensing*, 12, octobre 2020.

[6] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Esesn, Abdul A. S. Awwal et Vijayan K. Asari : The History Began from AlexNet : A Comprehensive Survey on Deep Learning Approaches. *arXiv :1803.01164 [cs]*, septembre 2018.

[7] Ethem Alpaydin : *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 2nd ed édition, 2010.

[8] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler et Bernt Schiele : 2D Human Pose Estimation : New Benchmark and State of the Art Analysis. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[9] Adel Bakhshipour, Abdolabbas Jafari, Seyed Mehdi Nassiri et Dariush Zare : Weed segmentation using texture features extracted from wavelet sub-images. *Biosystems Engineering*, 157:1–12, mai 2017.

[10] Suchet Bargoti et James Underwood : Deep Fruit Detection in Orchards. *arXiv :1610.03677 [cs]*, septembre 2017.

[11] Rodrigo Benenson, Stefan Popov et Vittorio Ferrari : Large-scale interactive object segmentation with human annotators. *arXiv :1903.10830 [cs]*, avril 2019.

[12] Christopher M. Bishop : *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

[13] Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao : YOLOv4 : Optimal Speed and Accuracy of Object Detection. *arXiv :2004.10934 [cs, eess]*, avril 2020.

[14] John Bongaarts : Human population growth and the demographic transition. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 364(1532): 2985–2990, octobre 2009.

[15] J. Bossu, Ch. Gée, G. Jones et F. Truchetet : Wavelet transform to discriminate between crop and weed in perspective agronomic images. *Computers and Electronics in Agriculture*, 65(1):133–143, janvier 2009.

[16] Léon Bottou : Stochastic Gradient Descent Tricks. *In* Grégoire Montavon, Geneviève B. Orr et Klaus-Robert Müller, éditeurs : *Neural Networks : Tricks of the Trade : Second Edition*, Lecture Notes in Computer Science, pages 421–436. Springer, Berlin, Heidelberg, 2012.

[17] Jean-yves Bouguet : Pyramidal implementation of the Lucas Kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

[18] Leo Breiman : Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[19] Leo Breiman : Statistical Modeling : The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3), août 2001.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever et Dario Amodei : Language Models are Few-Shot Learners. *arXiv :2005.14165 [cs]*, juillet 2020.

[21] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei et Yaser Sheikh : OpenPose : Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv :1812.08008 [cs]*, décembre 2018.

[22] Zhe Cao, Tomas Simon, Shih-En Wei et Yaser Sheikh : Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, Honolulu, HI, juillet 2017. IEEE.

[23] Toby N. Carlson et David A. Ripley : On the relation between NDVI, fractional vegetation cover, and leaf area index. *Remote Sensing of Environment*, 62(3):241–252, décembre 1997.

[24] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski et Armand Joulin : Emerging Properties in Self-Supervised Vision Transformers. *arXiv :2104.14294 [cs]*, avril 2021.

[25] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard et Cyrill Stachniss : Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *The International Journal of Robotics Research*, 36(10):1045–1052, septembre 2017.

[26] Keun Ha Choi, Sang Kwon Han, Sang Hoon Han, Kwang-Ho Park, Kyung-Soo Kim et Soohyun Kim : Morphology-based guidance line extraction for an autonomous weeding robot in paddy fields. *Computers and Electronics in Agriculture*, 113:266–274, avril 2015.

[27] Ronan Collobert et Samy Bengio : Links between perceptrons, MLPs and SVMs. *ICML*, 2004.

[28] Thomas H. Cormen et Thomas H. Cormen, éditeurs. *Introduction to Algorithms*. MIT Press, Cambridge, Mass, 2nd ed édition, 2001.

[29] Ingrid Daubechies : *Ten Lectures on Wavelets*. Numéro 61 de CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, Pa, 1992.

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei : ImageNet : A large-scale hierarchical image database. *In 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami, FL, juin 2009. IEEE.

[31] Andrei Dobrescu, Mario Valerio Giuffrida et Sotirios A. Tsaftaris : Doing More With Less : A Multitask Deep Learning Approach in Plant Phenotyping. *Frontiers in Plant Science*, 11:141, février 2020.

[32] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit et Neil Houlsby : An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale. octobre 2020.

[33] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang et Qi Tian : CenterNet : Keypoint Triplets for Object Detection. *arXiv :1904.08189 [cs]*, avril 2019.

[34] Mads Dyrmann, Henrik Karstoft et Henrik Skov Midtiby : Plant species classification using deep convolutional neural network. *Biosystems Engineering*, 151:72–80, novembre 2016.

[35] Paul F. Evangelista, Mark J. Embrechts et Boleslaw K. Szymanski : Taming the Curse of Dimensionality in Kernels and Novelty Detection. *In* Ajith Abraham, Bernard de Baets, Mario Köppen et Bertram Nickolay, éditeurs : *Applied Soft Computing Technologies : The Challenge of Complexity*, pages 425–438. Springer-Verlag, Berlin/Heidelberg, 2006.

[36] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn et Andrew Zisserman : The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, juin 2010.

[37] Gunnar Farnebäck : Two-Frame Motion Estimation Based on Polynomial Expansion. *In* Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Josef Bigun et Tomas Gustavsson, éditeurs : *Image Analysis*, volume 2749, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[38] Evelyn Fix et J. L. Hodges : Discriminatory Analysis. Nonparametric Discrimination : Consistency Properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238, décembre 1989.

[39] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi et Alexander C. Berg : DSSD : Deconvolutional Single Shot Detector. *arXiv :1701.06659 [cs]*, janvier 2017.

[40] Iván D. García-Santillán et Gonzalo Pajares : On-line crop/weed discrimination through the Mahalanobis distance from images in maize fields. *Biosystems Engineering*, 166:28–43, février 2018.

[41] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li et Jian Sun : YOLOX : Exceeding YOLO Series in 2021. *arXiv :2107.08430 [cs]*, août 2021.

[42] R. Gebbers et V. I. Adamchuk : Precision Agriculture and Food Security. *Science*, 327(5967):828–831, février 2010.

[43] Ross GIRSHICK : Fast R-CNN. *arXiv :1504.08083 [cs]*, avril 2015.

[44] Ross GIRSHICK, Jeff DONAHUE, Trevor DARRELL et Jitendra MALIK : Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *In 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, juin 2014.

[45] Mario Valerio GIUFFRIDA, Peter DOERNER et Sotirios A. TSAFTARIS : Pheno-Deep Counter : A unified and versatile deep learning architecture for leaf counting. *The Plant Journal*, 96(4):880–890, novembre 2018.

[46] G. GRENIER : *Agriculture de Précision : Comprendre et Mettre En Oeuvre Les Bases de La Révolution Agronomique.* Agriproduction. Univers Agricole. Editions France Agricole, 2018.

[47] H. W. GRIEPENTROG, M. NOERREMARK, J. NIELSEN et J. S. IBARRA : Autonomous Inter-Row Hoeing using GPS-based side-shift Control. *Agricultural Engineering International : CIGR Journal*, juillet 2007.

[48] Hans W. GRIEPENTROG, S. CHRISTENSEN, H.T. SØGAARD, Michael NØRREMARK, Ivar LUND et Enrico GRAGLIA : Robotic Weeding. *In AgEng 2004*, pages 12–16, septembre 2004.

[49] Nathan HAGEN et Michael W. KUDENOV : Review of snapshot spectral imaging technologies. *Optical Engineering*, 52(9):090901, septembre 2013.

[50] Stephen HANSON et Lorien PRATT : Comparing Biases for Minimal Network Construction with Back-Propagation. *In Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1989.

[51] C. HARRIS et M. STEPHENS : A Combined Corner and Edge Detector. *In Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, Manchester, 1988. Alvey Vision Club.

[52] Peter HART : How the Hough transform was invented. *IEEE Signal Processing Magazine*, 26(6):18–22, novembre 2009.

[53] Sebastian HAUG, Andreas MICHAELS, Peter BIBER et Jorn OSTERMANN : Plant classification system for crop /weed discrimination without segmentation. *In IEEE Winter Conference on Applications of Computer Vision*, pages 1142–1149, Steamboat Springs, CO, USA, mars 2014. IEEE.

[54] Sebastian HAUG et Jörn OSTERMANN : A Crop/Weed Field Image Dataset for the Evaluation of Computer Vision Based Precision Agriculture Tasks. *In* Lourdes

Agapito, Michael M. Bronstein et Carsten Rother, éditeurs : *Computer Vision - ECCV 2014 Workshops*, volume 8928, pages 105–116. Springer International Publishing, Cham, 2015.

[55] Kaiming He, Georgia Gkioxari, Piotr Dollár et Ross Girshick : Mask R-CNN. *In 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, octobre 2017.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun : Deep Residual Learning for Image Recognition. *In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, juin 2016. IEEE.

[57] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever et Ruslan R. Salakhutdinov : Improving neural networks by preventing co-adaptation of feature detectors. *arXiv :1207.0580 [cs]*, juillet 2012.

[58] Kurt Hornik, Maxwell Stinchcombe et Halbert White : Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, janvier 1989.

[59] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama et Kevin Murphy : Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297, juillet 2017.

[60] Weiting Huang, Pengfei Ren, Jingyu Wang, Qi Qi et Haifeng Sun : AWR : Adaptive Weighting Regression for 3D Hand Pose Estimation. *arXiv :2007.09590 [cs, eess]*, juillet 2020.

[61] Intergovernmental Panel on Climate Change, éditeur. *Climate Change 2013 - The Physical Science Basis : Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, 2014.

[62] Sergey Ioffe et Christian Szegedy : Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv :1502.03167 [cs]*, février 2015.

[63] Alan Julian Izenman : *Linear Discriminant Analysis*, pages 237–280. Springer New York, New York, NY, 2013.

[64] Hong Y. Jeon, Lei F. Tian et Heping Zhu : Robust Crop and Weed Segmentation under Uncontrolled Outdoor Illumination. *Sensors*, 11(6):6270–6283, juin 2011.

[65] Jiang, Wang, Zhuang, Li, Li et Gong : Leaf Counting with Multi-Scale Convolutional Neural Network Features and Fisher Vector Coding. *Symmetry*, 11(4):516, avril 2019.

[66] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng et Rong Qu : A Survey of Deep Learning-Based Object Detection. *IEEE Access*, 7:128837–128868, 2019.

[67] Gawain Jones, Christelle Gée et Frederic Truchetet : *Crop/Weed Discrimination in Simulated Images - Art. No. 64970E*, volume 6497. février 2007.

[68] Andreas Kamilaris et Francesc X. Prenafeta-Boldú : Deep learning in agriculture : A survey. *Computers and Electronics in Agriculture*, 147:70–90, avril 2018.

[69] Asifullah Khan, Anabia Sohail, Umme Zahoora et Aqsa Saeed Qureshi : A Survey of the Recent Architectures of Deep Convolutional Neural Networks. *Artificial Intelligence Review*, 53(8):5455–5516, décembre 2020.

[70] Diederik P. Kingma et Jimmy Ba : Adam : A Method for Stochastic Optimization. *arXiv :1412.6980 [cs]*, janvier 2017.

[71] Justin Kitzes, Mathis Wackernagel, Jonathan Loh, Audrey Peller, Steven Goldfinger, Deborah Cheng et Kallin Tea : Shrink and share : Humanity's present and future Ecological Footprint. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 363(1491):467–475, février 2008.

[72] Hans Knutsson : Representing local structure using tensors. *In Proceedings of 6th Scandinavian Conference on Image Analysis*, janvier 1989.

[73] Sven Kreiss, Lorenzo Bertoni et Alexandre Alahi : PifPaf : Composite Fields for Human Pose Estimation. *arXiv :1903.06593 [cs]*, avril 2019.

[74] Alex Krizhevsky, Ilya Sutskever et Geoffrey Hinton : ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, 25, janvier 2012.

[75] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig et Vittorio Ferrari : The Open Images Dataset V4 : Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981, juillet 2020.

[76] Louis Lac, Jean-Pierre Da Costa, Marc Donias, Barna Keresztes et Alain Bardet : Crop stem detection and tracking for precision hoeing using deep learning. *Computers and Electronics in Agriculture*, 192:106606, janvier 2022.

[77] Louis Lac, Jean-Pierre Da Costa, Marc Donias, Barna Keresztes et Louargant Marine : SDNet : Unconstrained Object Structure Detector Network for In-Field Real-Time Crop Part Location And Phenotyping. *In British Machine Vision Conference (BMVC)*, page 13, 2021.

[78] Louis Lac, Barna Keresztes, Marine Louargant, Marc Donias et Jean-Pierre Da Costa : An annotated image dataset of vegetable crops at an early stage of growth for proximal sensing applications. *Data in Brief*, page 108035, mars 2022.

[79] Pat Langley : The changing science of machine learning. *Machine Learning*, 82(3):275–279, mars 2011.

[80] Hei Law et Jia Deng : CornerNet : Detecting Objects as Paired Keypoints. *arXiv :1808.01244 [cs]*, août 2018.

[81] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner : Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, novembre 1998.

[82] Yann Lecun et Yoshua Bengio : Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361:1995, 1995.

[83] Wonyeol Lee, Hangyeol Yu, Xavier Rival et Hongseok Yang : On Correctness of Automatic Differentiation for Non-Differentiable Functions. *arXiv :2006.06903 [cs, stat]*, octobre 2020.

[84] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie : Feature Pyramid Networks for Object Detection. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, juillet 2017.

[85] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He et Piotr Dollár : Focal Loss for Dense Object Detection. *In 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, octobre 2017.

[86] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C. Lawrence Zitnick : Microsoft COCO : Common Objects in Context. *In* David Fleet, Tomas Pajdla, Bernt Schiele et Tinne Tuytelaars, éditeurs : *Computer Vision – ECCV 2014*, volume 8693, pages 740–755. Springer International Publishing, Cham, 2014.

[87] Zachary C. Lipton et Jacob Steinhardt : Troubling Trends in Machine Learning Scholarship. *arXiv :1807.03341 [cs, stat]*, juillet 2018.

[88] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu et Matti Pietikäinen : Deep Learning for Generic Object Detection : A Survey. *arXiv :1809.02165 [cs]*, août 2019.

[89] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi et Jiaya Jia : Path Aggregation Network for Instance Segmentation. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, juin 2018.

[90] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C. Berg : SSD : Single Shot MultiBox Detector. *In* Bastian Leibe, Jiri Matas, Nicu Sebe et Max Welling, éditeurs : *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 21–37, Cham, 2016. Springer International Publishing.

[91] Guillaume Lobet : Image Analysis in Plant Sciences : Publish Then Perish. *Trends in Plant Science*, 22(7):559–566, juillet 2017.

[92] Javier Garcia Lopez, Antonio Agudo et Francesc Moreno-Noguer : Vehicle pose estimation via regression of semantic points of interest. *In 2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 209–214, Dubrovnik, Croatia, septembre 2019. IEEE.

[93] Ilya Loshchilov et Frank Hutter : Decoupled Weight Decay Regularization. *arXiv :1711.05101 [cs, math]*, janvier 2019.

[94] Philipp Lottes, Jens Behley, Andres Milioto et Cyrill Stachniss : Fully Convolutional Networks With Sequential Information for Robust Crop and Weed Detection in Precision Farming. *IEEE Robotics and Automation Letters*, 3(4):2870–2877, octobre 2018.

[95] David G. Lowe : Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, novembre 2004.

[96] Warren S. McCulloch et Walter Pitts : A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, décembre 1943.

[97] Chenyong Miao, Alice Guo, Jinliang Yang, Yufeng Ge et James C. Schnable : Automation of Leaf Counting in Maize and Sorghum Using Deep Learning. Preprint, Plant Biology, décembre 2020.

[98] Andres Milioto, Philipp Lottes et Cyrill Stachniss : Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. *In 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2229–2235, mai 2018.

[99] Mehryar Mohri, Afshin Rostamizadeh et Ameet Talwalkar : *Foundations of Machine Learning*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, 2012.

[100] M. Montalvo, G. Pajares, J.M. Guerrero, J. Romeo, M. Guijarro, A. Ribeiro, J.J. Ruz et J.M. Cruz : Automatic detection of crop rows in maize fields with high weeds pressure. *Expert Systems with Applications*, 39(15):11889–11897, novembre 2012.

[101] David J. Mulla : Twenty five years of remote sensing in precision agriculture : Key advances and remaining knowledge gaps. *Biosystems Engineering*, 114(4):358–371, avril 2013.

[102] Alejandro Newell, Zhiao Huang et Jia Deng : Associative Embedding : End-to-End Learning for Joint Detection and Grouping. *arXiv :1611.05424 [cs]*, novembre 2016.

[103] Alejandro Newell, Kaiyu Yang et Jia Deng : Stacked Hourglass Networks for Human Pose Estimation. *arXiv :1603.06937 [cs]*, mars 2016.

[104] Nobuyuki Otsu : A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, janvier 1979.

[105] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto et Eduardo A. B. da Silva : A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3):279, janvier 2021.

[106] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller et Vittorio Ferrari : We don't need no bounding-boxes : Training object class detectors using only human verification. *arXiv :1602.08405 [cs]*, avril 2017.

[107] Luis Perez et Jason Wang : The Effectiveness of Data Augmentation in Image Classification using Deep Learning. décembre 2017.

[108] Francis J. Pierce et Peter Nowak : Aspects of Precision Agriculture. *In* Donald L. Sparks, éditeur : *Advances in Agronomy*, volume 67, pages 1–85. Academic Press, janvier 1999.

[109] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen et S. S. Iyengar : A Survey on Deep Learning : Algorithms, Techniques, and Applications. *ACM Computing Surveys*, 51(5):1–36, janvier 2019.

[110] Maryam Rahnemoonfar et Clay Sheppard : Deep Count : Fruit Counting Based on Deep Simulated Learning. *Sensors*, 17(4):905, avril 2017.

[111] Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi : You Only Look Once : Unified, Real-Time Object Detection. *In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, juin 2016.

[112] Joseph Redmon et Ali Farhadi : YOLO9000 : Better, Faster, Stronger. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, juillet 2017.

[113] Joseph Redmon et Ali Farhadi : YOLOv3 : An Incremental Improvement. *arXiv :1804.02767 [cs]*, avril 2018.

[114] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, juin 2017.

[115] I. F. Rodríguez, K. Branson, E. Acuña, J. L. Agosto-Rivera, T. Giray et R. Mégret : Honeybee Detection and Pose Estimation using Convolutional Neural Networks. *Congrès Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP)*, juin 2018.

[116] Olaf Ronneberger, Philipp Fischer et Thomas Brox : U-Net : Convolutional Networks for Biomedical Image Segmentation. *arXiv :1505.04597 [cs]*, mai 2015.

[117] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov et Liang-Chieh Chen : MobileNetV2 : Inverted Residuals and Linear Bottlenecks. janvier 2018.

[118] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas et Aleksander Madry : How Does Batch Normalization Help Optimization ? *arXiv :1805.11604 [cs, stat]*, avril 2019.

[119] Gilbert Saporta et Gérard Hatabian : Régions de confiance en analyse factorielle. *In Fourth International Symposium on Data Analysis and Informatics*, volume IV, pages 499–508. Elsevier Science Publishers, 1986.

[120] Lalit Saxena et Leisa Armstrong : A survey of image processing techniques for agriculture. *Australian Society of Information and Communication Technologies in Agriculture*, page 14, 2014.

[121] Shaohuai Shi, Qiang Wang, Pengfei Xu et Xiaowen Chu : Benchmarking State-of-the-Art Deep Learning Software Tools. *In 2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104, novembre 2016.

[122] Karen Simonyan et Andrew Zisserman : Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv :1409.1556 [cs]*, septembre 2014.

[123] Søren Skovsen, Mads Dyrmann, Anders Krogh Mortensen, Kim Arild Steen, Ole Green, Jørgen Eriksen, René Gislum, Rasmus Nyholm Jørgensen et Henrik Karstoft : Estimation of the Botanical Composition of Clover-Grass Leys from RGB Images Using Data Simulation and Fully Convolutional Neural Networks. *In Sensors*, 2017.

[124] Emma Strubell, Ananya Ganesh et Andrew McCallum : Energy and Policy Considerations for Deep Learning in NLP. *arXiv :1906.02243 [cs]*, juin 2019.

[125] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang et Joel S. Emer : Efficient Processing of Deep Neural Networks : A Tutorial and Survey. *Proceedings of the IEEE*, 105(12): 2295–2329, décembre 2017.

[126] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke et Andrew Rabinovich : Going Deeper with Convolutions. *arXiv :1409.4842 [cs]*, septembre 2014.

[127] Lisa Torrey et Jude Shavlik : *Transfer Learning.* 2009.

[128] Sotirios A. Tsaftaris, Massimo Minervini et Hanno Scharr : Machine Learning for Plant Phenotyping Needs Image Processing. *Trends in Plant Science*, 21(12):989–991, décembre 2016.

[129] Vladimir N Vapnik : *The Nature of Statistical Learning Theory.* Springer New York : Imprint : Springer, New York, NY, 2000.

[130] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser et Illia Polosukhin : Attention Is All You Need. *arXiv :1706.03762 [cs]*, juin 2017.

[131] M. Verucchi, G. Brilli, D. Sapienza, M. Verasani, M. Arena, F. Gatti, A. Capotondi, R. Cavicchioli, M. Bertogna et M. Solieri : A Systematic Assessment of Embedded Neural Networks for Object Detection. *In 2020 25th*

IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 937–944, septembre 2020.

[132] Chien-Yao Wang, Alexey Bochkovskiy et Hong-Yuan Mark Liao : Scaled-YOLOv4 : Scaling Cross Stage Partial Network. *arXiv :2011.08036 [cs]*, novembre 2020.

[133] Chien-Yao Wang, I.-Hau Yeh et Hong-Yuan Mark Liao : You Only Learn One Representation : Unified Network for Multiple Tasks. *arXiv :2105.04206 [cs]*, mai 2021.

[134] Alexander Wendel et James Underwood : Self-supervised weed detection in vegetable crops using ground based hyperspectral imaging. *In 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5128–5135, Stockholm, Sweden, mai 2016. IEEE.

[135] D. M. Woebbecke, G. E. Meyer, K. Von Bargen et D. A. Mortensen : Color Indices for Weed Identification Under Various Soil, Residue, and Lighting Conditions. *Transactions of the ASAE*, 38(1):259–269, 1995.

[136] Xiaolong Wu, Stéphanie Aravecchia, Philipp Lottes, Cyrill Stachniss et Cédric Pradalier : Robotic weed control using automated weed and crop classification. *Journal of Field Robotics*, 37(2):322–340, mars 2020.

[137] Bin Xiao, Haiping Wu et Yichen Wei : Simple Baselines for Human Pose Estimation and Tracking. *arXiv :1804.06208 [cs]*, avril 2018.

[138] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li et Ping Luo : DetCo : Unsupervised Contrastive Learning for Object Detection. février 2021.

[139] Weihui Zeng et Miao Li : Crop leaf disease recognition based on Self-Attention convolutional neural network. *Computers and Electronics in Agriculture*, 172:105341, mai 2020.

[140] Dimitris Zermas, Vassilios Morellas, David Mulla et Nikos Papanikolopoulos : 3D model processing for high throughput phenotype extraction – the case of corn. *Computers and Electronics in Agriculture*, page 105047, novembre 2019.

[141] Naiqian Zhang, Maohua Wang et Ning Wang : Precision agriculture—a worldwide overview. *Computers and Electronics in Agriculture*, 36(2):113–132, novembre 2002.

[142] Xingyi Zhou, Dequan Wang et Philipp Krähenbühl : Objects as Points. *arXiv :1904.07850 [cs]*, avril 2019.

[143] Xingyi Zhou, Jiacheng Zhuo et Philipp Krähenbühl : Bottom-up Object Detection by Grouping Extreme and Center Points. *arXiv :1901.08043 [cs]*, avril 2019.

[144] Zhengxia Zou, Zhenwei Shi, Yuhong Guo et Jieping Ye : Object Detection in 20 Years : A Survey. *arXiv :1905.05055 [cs]*, mai 2019.

[145] P Zwaenepoel et J M Le Bars : L'agriculture de précision. *CEMAGREF*, page 15, 1997.

Méthodes de vision par ordinateur et d'apprentissage profond pour la localisation, le suivi et l'analyse de la structure de plantes. Application au désherbage de précision.

Résumé :
L'agriculture fait aujourd'hui face au défi de la constante augmentation des besoins alimentaires, causée par la croissance démographique soutenue. En parallèle, la crise environnementale, dont l'agriculture est en partie responsable, risque d'impacter négativement les sociétés humaines. Limiter ces impacts tout en assurant la sécurité alimentaire mondiale est donc une nécessité. L'agriculture de précision visant à traiter les cultures de manière plus fine et plus localisée que les méthodes traditionnelles de l'agriculture intensive est l'un des leviers de la transition vers une agriculture plus résiliente et moins polluante. Les récentes avancées en vision par ordinateur, en robotique et en intelligence artificielle permettent d'imaginer de nouvelles solutions innovantes afin d'automatiser certaines tâches comme le désherbage de précision.

Dans ce travail de thèse, des méthodes de vision par ordinateur et d'apprentissage profond sont explorées afin de détecter et de collecter des propriétés sur des plantes d'intérêt ainsi que pour suivre des plantes dans des séquences d'images. D'abord, ce travail propose d'employer un réseau de détection d'objets dans le but de localiser précisément les organes des plantes et d'estimer leur nombre de feuilles. Ensuite, il détaille un réseau de neurones profond, inspiré des détecteurs de pose, conçu pour détecter la structure non contrainte des plantes, c'est-à-dire la position des organes et leurs relations. Finalement, ce travail propose un algorithme de suivi de tiges de plantes dans des séquences d'images afin d'améliorer la robustesse de la détection des plantes.

Le travail proposé s'appuie sur un prototype de bloc-outil de désherbage mécanique de précision, nommé BIPBIP, pour évaluer expérimentalement les méthodes développées. Deux cultures en plein champ à un stade précoce de développement sont envisagées : le maïs et le haricot. Des expérimentations sont menées afin de paramétrer, d'évaluer et de comparer les méthodes proposées. Elles montrent qu'il est possible, d'une part, d'exploiter les réseaux de neurones profonds afin de détecter la structure des plantes et de localiser précisément leurs organes et, d'autre part, d'améliorer la performance via un algorithme de suivi temporel des détections.

Mots-clés : vision par ordinateur, apprentissage profond, agriculture de précision, désherbage de précision.

Unité de recherche :
UMR 5218 Université de Bordeaux, 33000 Bordeaux, France.

Computer vision and deep learning for locating, tracking and analyzing the structure of plants. Application to precision hoeing.

Abstract:
Agriculture faces today the challenge of steadily increasing food demands, caused by the sustained population growth. At the same time, the environmental crisis, for which agriculture is partly responsible, risks having a negative impact on human societies. Limiting these impacts while ensuring global food security is therefore a necessity. Precision agriculture, which aims to treat crops in a more precise and more localized way than traditional methods of intensive agriculture, is one of the levers of the transition to a more resilient and less polluting agriculture. Recent advances in computer vision, robotics and artificial intelligence make it possible to imagine new innovative solutions to automate certain tasks such as precision hoeing.

In this thesis work, computer vision and deep learning methods are explored in order to detect and collect properties on plants of interest as well as to track crops in sequences of images. First, this work proposes to use an object detection neural network in order to detect the precise location of plant organs and to regress their number of leaves. Second, it details a deep neural architecture inspired by pose estimation neural networks to detect the unconstrained structure of plants, i.e. the location of organs and their relations. Finally, this work proposes an algorithm for tracking plant stems in sequences of images in order to improve the robustness of the detection of plants.

The proposed work is based on a mechanical precision weeding tool prototype named BIPBIP to experimentally evaluate the developed methods. Two crops at an early stage of development are targeted: maize and bean. Several experiments are carried out to parameterize, evaluate and compare the proposed methods. They show that it is possible, on the one hand, to leverage deep neural networks to detect the structure of crops and to precisely locate their organs and, on the other hand, to improve the performance of the detection thanks to a temporal tracking algorithm.

Keywords: computer vision, deep learning, precision agriculture, precision hoeing.

Search unit:
UMR 5218 Université de Bordeaux, 33000 Bordeaux, France.