Analyse d'images en élevage aviaire

Henry Brunet

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Henry BRUNET

Le 22 juillet 2022

Analyse d'images en élevage aviaire

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Mathématiques et Applications**

Unité de recherche :
IMT : Institut de Mathématiques de Toulouse

Thèse dirigée par
Didier CONCORDET

Jury

M. Thomas LALOë, Rapporteur
M. Peggy GANDIA, Examinateur
M. Didier CONCORDET, Directeur de thèse
M. Jérôme SARACCO, Président

Remcerciements

Je tiens tout d'abord à remercier mon directeur de thèse Monsieur Didier Concordet pour sa disponibilité permanente et pour les nombreux conseils qu'il m'a prodigués.

Je remercie l'ensemble des personnes de l'ITAVI et particulièrement Pauline Creach pour sa disponibilité et surtout sa patience lors de nos nombreux échanges.

Je remercie également tous ceux qui ont bien voulu participer aux tâches ingrates, telles que devoir encadrer des poulets sur des images pendant des heures et relire mon manuscrit de thèse. Je remercie donc en premier lieu mes parents, ainsi que Mathieu, Anaïs, Valentine, Antoine, Hugue et Charles.

Je terminerai par Mehdi. Un grand merci à toi qui as su me conseiller, me soutenir et surtout me supporter sur l'ensemble de ces trois ans.

Table des matières

Table des figures

Liste des tableaux

List of Abbreviations

MOT	**M**ulti **O**bjects **T**racking
DL	**D**eep **L**earning
ML	**M**achine **L**earning
CNN	**C**onvolutional **N**eural **N**etwork
RPN	**R**egion **P**roposal **N**etwork
FPN	**F**eatures **P**yramid **N**etwork
YOLO	**Y**ou **O**nly **L**ook **O**nce
NMS	**N**on **M**aximum **S**uppression
RoI	**R**egion of **I**nterest
SIFT	**S**cale **I**nvarient **F**eatures **T**ransform
MHT	**M**ultiple **H**ypothesis **T**racking
JPDAF	**J**oin **P**robabilistic **D**ata **A**ssociation **F**iltering
IoU	**I**ntersection **o**ver **U**nion

Chapitre 1

Introduction

1.1 Problématique et contexte industriel

L'élevage de poulets de chair fait face à une demande de transparence de plus en plus grande de la part de la société afin de garantir aux consommateurs un élevage respectueux du bien-être animal. Connaître le comportement d'un animal peut donner une information sur son état de santé. Il est par exemple très intéressant d'un point de vue économique de détecter le plus rapidement possible une maladie qui se propage dans un élevage. Connaître le comportement d'un animal peut également donner une information sur son bien-être en général. De nombreux travaux visent aujourd'hui à évaluer ce comportement le plus précisément possible. Cette thèse CIFRE entre dans le cadre d'un projet appelé E-Broiler Track porté par l'ITAVI (L'institut technique des filières avicole, cunicole et piscicole) en partenariat avec INRAe et l'IDELE (institut de l'élevage). L'élevage de poulets de chair fait face à une demande de transparence de plus en plus grande de la part de la société afin de garantir aux consommateurs un élevage respectueux du bien-être animal. Connaître le comportement d'un animal peut donner une information sur son état de santé. Il est par exemple très intéressant d'un point de vue économique de détecter le plus rapidement possible une maladie qui se propage dans un élevage. Connaître le comportement d'un animal peut également apporter une information sur son bien-être en général. De nombreux travaux visent aujourd'hui à évaluer ce comportement le plus précisément possible. Cette thèse CIFRE entre dans le cadre d'un projet appelé E-Broiler Track porté par l'ITAVI (L'institut technique des filières avicoles, cunicoles et piscicoles) en partenariat avec INRAe et l'IDELE (institut de l'élevage).E-Broiler Track est un projet qui vise à développer un outil pour l'élevage de précision qui au moyen de caméras permet de mesurer des indicateurs de santé et de bien-être des animaux. Des méthodes d'évaluation du bien-être de poulets de chair, comme l'utilisation de puces RFID ((SIEGFORD et al., 2016), (SALES et al., 2015), (FEIYANG et al., 2016)) permettent de collecter toutes sortes de données (temps passés aux mangeoires, distances parcourues, ...). Mais son utilisation en élevage commercial n'est pas vraiment envisageable, car beaucoup trop chronophage pour un éleveur (notamment le retrait de la puce) et beaucoup trop onéreuse. La méthode EBENE® développée par l'ITAVI ((PAULINE et al., 2019)), nécessitent du temps et la présence d'un observateur formé. Cette méthode consiste à estimer le bon fonctionnement et le bien-être des animaux dans un élevage. Quoiqu'il n'y ait pas encore aujourd'hui

de définitions claires sur ce que l'on appelle le bien-être animal, on estime que celui-ci se trouve dans une condition de bien-être s'il peut librement s'exprimer (manger, battre des ailes, interagir avec d'autres poulets, etc.). Le bien-être est actuellement défini autour de quatre principes : la bonne alimentation, la bonne santé, un bon environnement et l'extériorisation de certains comportements d'intérêts. Ces principes sont respectés soit grâce à la mise en place de moyens adaptés à un bon élevage soit à travers un bon comportement de l'ensemble des poulets. Ces moyens sont par exemple la mise en place de matériels tels que des mangeoires, des abreuvoirs ou des perchoirs qui permettent alors aux poulets de s'exprimer. Ce peut être une litière régulièrement entretenue ou le passage régulier d'un vétérinaire. Cette thèse est tournée principalement vers la deuxième catégorie, c'est-à-dire le comportement de l'ensemble des poulets. Plusieurs critères comportementaux sont définis afin de définir un bon comportement animal. Parmi ces critères, on retrouve notamment : l'espace autour de chaque poulet, le pourcentage de poulets en mouvement/repos, le nombre de poulets aux mangeoires, leur réaction face au passage de l'éleveur, les bains de poussières ou bien encore les interactions positives/négatives (coups de bec) entre eux. La méthode EBENE® développée par l'ITAVI est une méthode qui permet de donner un score de bien-être à travers l'évaluation de tous ces critères. Pour cela, une personne préalablement formée à l'utilisation de cet outil (la méthode EBENE® a été traduite en application mobile) visite alors régulièrement l'élevage et évalue le bien-être des animaux. Cette personne sélectionne arbitrairement une zone de l'élevage de quelques mètres carrés et recense le nombre d'animaux couchés/debout, aux mangeoires, ainsi que tous les autres critères mentionnés précédemment afin de retourner le score de bien-être. Il est à noter que cette méthode n'est reliée à aucune réglementation nationale ou européenne. Celle-ci a seulement pour but de venir en aide aux éleveurs dans leur quotidien. Une telle méthode d'évaluation est limitée à un nombre assez réduit d'animaux observés et nécessite l'intervention d'une personne qualifiée, sans compter que deux personnes peuvent avoir des résultats différents. Enfin, l'intervention d'une personne dans l'élevage peut perturber le comportement des animaux. L'idée de ce projet est donc de développer un outil qui soit capable de mesurer des indicateurs comportementaux des animaux, de façon non intrusive, et en continu.

L'analyse d'image permet justement de surveiller un élevage en continue, et ce, de façon non intrusive. L'analyse d'images permet de détecter les poulets et de générer une grande partie des indicateurs mentionnés précédemment.

1.2 L'imagerie en milieu aviaire

Le fait est que plus la caméra est proche des animaux, plus la résolution de l'image est bonne et l'information captée précise. En revanche, une caméra éloignée permet de porter l'étude sur un plus grand nombre d'animaux. Ce sont les deux types d'études que l'on retrouve généralement aujourd'hui. Une vision rapprochée correspond généralement à une étude dans un cadre contrôlé, c'est-à-dire un petit espace fermé avec un faible nombre de poulets et une caméra à une distance de l'ordre du mètre des poulets. En revanche, les visions en champ large correspondent généralement à des prises de vues en élevage, en condition plus réelle.

Les méthodes que l'on retrouve chez (GUO et al., 2020), (VAN HERTEM et al., 2018), (NÄÄS et al., 2012) et (NEVES et al., 2015), analysent la distribution des animaux. Les caméras sont généralement placées aux plafonds des élevages, couvrant alors un grand champ de vision. Ces méthodes cherchent les zones de l'élevage occupées par les poulets, soit en divisant le champ de la caméra en sous-régions et en analysant l'intensité des pixels dans ces régions, soit en essayant de détecter directement chaque poulet en appliquant des seuils sur les pixels pour les distinguer du sol. En réalité, ces dernières méthodes fonctionnent sur des amas de poulets, car il est très compliqué de discerner chaque poulet en utilisant simplement des seuils sur des pixels. Ces analyses permettent de détecter par exemple des problèmes liés aux mangeoires, aux abreuvoirs ou tout simplement de détecter l'activité générale des poulets simplement en évaluant la variation de l'intensité des pixels.

Des méthodes telles que (AMRAEI, ABDANAN MEHDIZADEH et SALLARY, 2017) et (MOLLAH et al., 2010), permettent d'estimer le poids des animaux en fonction de leur surface perçue par la caméra. Ces tests sont effectués dans un cadre contrôlé (caméra proche et sur un faible nombre d'animaux).

La méthode de (PEREIRA et al., 2013) consiste à détecter des poulets isolés et à analyser leur silhouette. Grâce aux différents critères tels que la surface ou l'allongement de la silhouette, et grâce à l'élaboration d'un arbre de décision, la méthode permet de déterminer le type d'activité des poulets. Ces différentes activités peuvent être : un battement d'ailes, du picotage ou du grattage. L'analyse de ces activités peut donner une information sur le bien-être de l'animal.

Suivre les poulets permet d'avoir une information sur le comportement de l'animal de manière plus précise. C'est ce que fait (SERGEANT, BOYLE et FORBES, 1998) qui applique un suivi sur un petit groupe de poulet. Toutes les méthodes mentionnées jusqu'à présent analysent l'intensité des pixels afin de pouvoir distinguer d'une part les poulets et de l'autre le sol. Ces méthodes sont sensibles à la luminosité, au type de sol, et ne peuvent pas discerner les différents poulets lorsqu'ils sont regroupés en amas de plus de 6-7 animaux, ce qui est assez limitant pour un suivi à plus grande échelle.

Certains n'hésitent pas, comme (COLLINS, 2008) à relever manuellement la position des centres des poulets ainsi que de leur tête. Ce genre de traitement est effectué sur une vingtaine d'animaux sur une durée de dix minutes avec une image toutes les 5 secondes. Ces positions permettent par la suite d'analyser les trajectoires, les temps de déplacement, les vitesses moyennes des poulets ou la distance au plus proche voisin.

D'autres études plus récentes cherchent à suivre les poulets avec des méthodes de tracking par apprentissage profond. Ces méthodes de tracking nécessitent une très grande base de données et ne s'appliquent que sur de petites images. Ces différentes études sont toutes faites sur des zones contrôlées. C'est le cas de (FANG et al., 2020) (JU et al., 2020). Ces méthodes sont sensibles à l'initialisation du tracking. Elles permettent néanmoins de connaître la position et les déplacements de quelques poulets durant quelques minutes.

Enfin, d'autres méthodes comme (FANG et al., 2021) (ZHUANG et al., 2018) se basent sur des traits un peu plus précis de l'animal. Ils détectent les positions des pattes, du bec, etc.. Cela permet de détecter le même type de comportement que

ceux de (PEREIRA et al., 2013) , ou bien de détecter un animal malade si son comportement est inhabituel. Puisque l'on a besoin d'avoir accès à des traits fins de l'animal, l'étude se porte également sur une petite zone au sol, donc sur un très faible nombre d'animaux.

Les méthodes de tracking par apprentissage profond sont récentes. Comme l'explique (CIAPARRONE et al., 2020), il y a encore beaucoup de recherches à effectuer dans ce domaine.

1.3 Objectifs

L'idéal serait donc de pouvoir analyser le maximum d'informations possibles sur un poulet, appliqué à l'ensemble d'un élevage. Mais comme vu précédemment, il y a un compromis à faire entre le nombre de poulets à étudier et la finesse des critères que l'on cherche à détecter pour chaque poulet. En d'autres mots, soit l'on utilise peu de caméras placées très haut afin de couvrir un maximum de surface et auquel cas il devient très difficile de discerner deux poulets côte à côte, soit l'on utilise des caméras à basse hauteur afin d'avoir une résolution d'image la plus grande possible et auquel cas il est possible de détecter des bains de poussières et des picotages entre poulets. Le souci du deuxième cas de figure, c'est qu'il faudrait une centaine de caméras pour couvrir un élevage, ce qui entraînerait des coûts de matériel et de calculs importants. À noter qu'il ne peut être possible d'analyser qu'une partie de l'élevage, mais celui-ci doit être représentatif de l'ensemble de l'élevage ; et plus la surface à analyser est petite, plus il est difficile de pouvoir reporter les données à l'ensemble d'un bâtiment.

Il faut donc développer un outil qui puisse retourner des indicateurs aux plus proches de ceux de la méthode EBENE® tout en étant applicable à un maximum de poulets. Il faut également tirer profit des avantages que fournit une analyse par vidéo. Il est possible grâce à l'analyse vidéo de créer des indicateurs qui ne pouvaient pas être mesurés par un simple regard d'un observateur. Contrairement à l'application développée à partir de la méthode EBENE® qui retourne directement un score de bien-être, cet outil n'a pour l'instant pas pour objectif de retourner un score, mais simplement de retourner un maximum d'indicateurs. Puisque le bien-être animal est une notion difficile à quantifier, il faut également pouvoir dire ce qui fait qu'un indicateur traduise un état de bien-être ou non. Cette étude fait partie des suites de ce projet.

L'analyse vidéo choisie pour ce projet consiste à détecter et suivre individuellement chaque poulet sous le champ de la caméra. Le tracking consiste à suivre la position d'un ou plusieurs objets sur l'ensemble des images d'une vidéo. Il faut donc être capable d'une part de détecter ces objets sur chaque image de la vidéo, et d'autre part de mettre en correspondance ces mêmes objets que l'on retrouve d'une image à l'autre. Il existe aujourd'hui des méthodes capables d'effectuer ces deux actions (détection et association de données) en même temps, et celles qui les font séparément. Parmi ces méthodes on retrouve également celles basées sur de l'apprentissage, et celles basées sur des techniques classiques de traitement d'images. Globalement les

méthodes par apprentissage profond (**DL** pour **D**eep **L**earning) atteignent aujourd'hui d'excellents taux de réussite de tracking, et ces modèles ne cessent de s'améliorer. Le tableau 1.1 donne un exemple d'articles présentant différentes méthodes de tracking. Suivre les poulets apporte bien plus d'informations que d'estimer leur densité de présence. Cela permet de connaître les temps passés aux mangeoires et aux abreuvoirs, de connaître les distances parcourues, leurs vitesses ou simplement leur temps de repos. Une telle connaissance passe par une bonne détection et un bon algorithme de suivi des animaux. Actuellement la meilleure solution semble être d'utiliser une méthode d'apprentissage profond pour gérer la détection des poulets. Cela permet d'être moins sensible aux variations d'éclairement, mais surtout de pouvoir identifier les poulets dans les regroupements, ce qui reste une vraie limite à la détection de poulets par des méthodes de seuil sur des intensités aussi évoluées soient-elles. Les méthodes de tracking par apprentissage profond manquent encore de performance pour un tel usage, et également d'algorithmes accessibles au public. C'est pourquoi le suivi doit se tourner vers une méthode plus " classique " . Comme présenté précédemment, il existe à ce jour plusieurs recherches sur la détection et le suivi de poulets à travers des images de vidéos surveillances, mais principalement en environnement expérimental. À notre connaissance, il n'existe pas à ce jour d'étude qui vise à générer des indicateurs comportementaux aussi complets, à partir d'images de vidéos surveillances en élevage commercial. Les indicateurs que l'on mesure sont principalement :

- **Le temps passé aux mangeoires.** Une zone circulaire autour de chaque mangeoire a été estimée expérimentalement. Un animal dont le centre est enregistré à l'intérieur de cette zone est considéré comme ayant accès à la mangeoire. On connaît donc pour chaque poulet, le temps et la fréquence à laquelle celui-ci se rend à une mangeoire.
- **Le temps passé aux abreuvoirs.** De la même façon que pour les mangeoires, des zones ont été estimées autour des abreuvoirs, à l'intérieur desquelles un poulet est considéré comme ayant accès à l'abreuvoir.
- **Les distances parcourues.** Connaissant la position de chaque animal au cours du temps, il est alors possible de mesurer les distances parcourues pour chaque animal. Celles-ci sont reportées en centimètres.
- **Les vitesses atteintes.** Tout comme pour les distances mesurées précédemment, il est possible de mesurer les vitesses (nombre de pixels par image). Celles-ci sont traduites en centimètres par seconde.
- **Les surfaces.** La surface de chaque poulet se traduit à travers la surface du rectangle circonscrit à l'animal. Une analyse a été portée sur la relation qui existe entre la surface de ce rectangle et le poids des animaux.
- **Les espaces disponibles.** L'espace disponible de chaque animal est représenté par la surface de sa cellule de Voronoï (cellules dont les arêtes correspondent aux médianes entre chaque poulet).
- **Espace visité.** Cet indicateur est un complément de celui sur les distances parcourues. Cet indicateur traduit le côté explorateur du poulet, à savoir si ses déplacements sont concentrés au même endroit de l'image ou pas. Pour cela l'image est découpée en 256 zones (valeur déterminée expérimentalement). Puis l'on relève le nombre de cases visitées.

— **L'activité.** Un poulet est considéré actif lorsque celui-ci se déplace ; et inactif lorsque sa position ne bouge pas. Pour cela un lissage est effectué sur ses positions et un seuillage de déplacement a été estimé. Pour tout déplacement supérieur à ce seuil, l'animal est considéré comme en déplacement. Ce lissage et ce seuil permettent de s'affranchir des variations de mouvement dues à l'imprécision de la détection.
itemizeitemizeitemize

TABLE 1.1 – Classement des méthodes de tracking

	étape unique	**détection + association de données**	
		détection	**association de données**
méthodes par apprentissage	(JIANG et al., 2019) (FUCHS et al., 2019)	(DHILLON et VERMA, 2020) (MITTAL, SRIVASTAVA et CHAWLA, 2019)	(YOON et al., 2019) (ILTIS et TING, 1991)
méthodes classiques de traitement d'images	(ZHANG et VAN DER MAATEN, 2013)	(HU, BARNARD et COLLOMOSSE, 2010) (LOWE, 2004)	(MUNKRES, 1957) (BAR-SHALOM, DAUM et HUANG, 2009)
	(YANG, YU et WU, 2007)	(BALAJI et KARTHIKEYAN, 2017) (JOSHI et THAKORE, 2012)	

1.4 Déroulement du projet et plan

Grâce à une première série de captations vidéo dans un environnement contrôlé à INRAe de Nouzilly, il a été possible de déterminer la meilleure façon d'effectuer ce tracking individuel de poulets de chair. S'en est suivie une série de captations en élevage commercial afin de déterminer la meilleure façon d'utiliser les caméras pour une acquisition optimale des données. Ces tests en élevage commercial ont également permis de paramétrer et de tester l'outil de tracking de poulets de chair développé lors de ce projet. Enfin, il a été déterminé la meilleure façon de transformer ces données brutes de tracking afin de créer des indicateurs pertinents du bien-être animal. La première partie de ce manuscrit présente les différentes méthodes de *détections* envisagées, détaille plus précisément celle qui a été retenue pour le tracking, et présente la façon dont les captations ont été faites. La deuxième partie du manuscrit se focalise sur la partie *suivi* des poulets. Là aussi, un état de l'art présente les différentes méthodes existantes et la méthode retenue est présentée plus

en détail. Enfin, la troisième partie de ce manuscrit présente les résultats des tests de détection et de suivi, effectués sur différentes bases de données. La dernière partie des résultats donne un aperçu des données qui peuvent être générées pour chaque type d'indicateurs.

Chapitre 2

La détection

2.1 Bibliographie

2.1.1 Présentation générale

2.1.1.1 Définitions

La détection d'objet permet de localiser l'instance d'un objet dans une image. Il faut donc être capable de limiter de façon spatiale l'objet dans l'image, et de reconnaître la classe (ou le type) d'objet à laquelle il appartient. La détection d'objets sur une image passe généralement par plusieurs étapes (Figure 2.1).

— **La segmentation** permet de partitionner l'image en régions homogènes. Cette étape permet de découper l'image en sous-régions appelées *imagettes*, chacune correspondant à une classe d'objets. À ce niveau-là, si deux objets du même type sont juxtaposés, il n'est pas rare de les voir regroupés sous une même imagette.

— **La classification** permet de dire à quelle classe appartient chaque imagette. Cela permet dans le cas simple de différencier l'arrière-plan des objets. Si plusieurs classes d'objets sont présentes dans l'image, la classification permet alors de distinguer ces objets entre eux.

— **La localisation** permet enfin de découper plusieurs objets appartenant à une même imagette. Cette étape retourne généralement les coordonnées du pixel central de chaque objet ainsi que les dimensions de ces derniers.

Toute l'information nécessaire à l'exécution de ces trois opérations est incluse dans l'intensité des pixels qui composent l'image, et dans la façon dont ils sont répartis. Il faut donc dans un premier temps, être capable d'extraire des images de l'information utile qui permette de partitionner l'image en autant de régions qu'il y a d'objets. Nous appelons ces informations utiles *features*. Ce mot très largement utilisé dans la littérature est ici défini comme toute information permettant d'identifier un objet. Certaines features sont accessibles directement en étudiant l'intensité des pixels d'une image comme la couleur. D'autres nécessitent un prétraitement de l'image qui retourne des *zones d'intérêts* de l'image. Ces zones d'intérêts peuvent par exemple être des contours d'objets. Ces zones d'intérêts font partie de la famille des features et permettent l'apparition de nouvelles features telle que le squelette d'un objet. On peut retrouver dans la littérature les termes de features locales pour les zones d'intérêts et de features globales pour les autres. Dans un deuxième

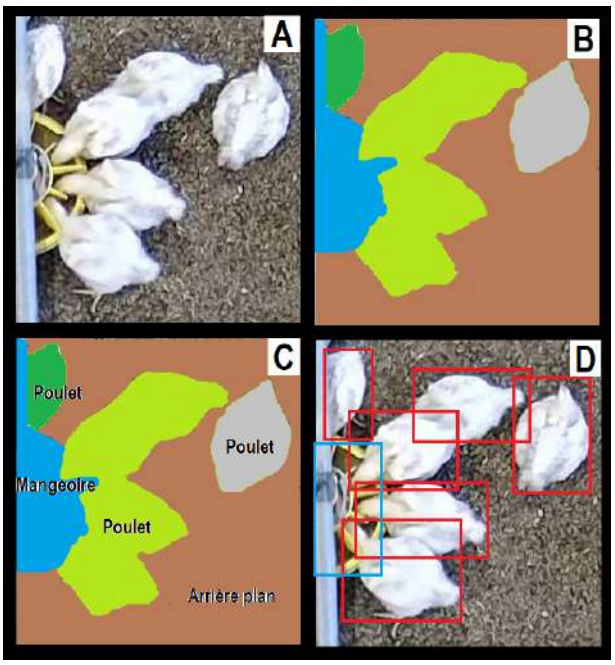

FIGURE 2.1 – A : Image initiale, B : Segmentation, C : Classification, D : localisation

temps, il faut pouvoir discriminer les différentes classes d'objets segmentées. Il est alors nécessaire de traduire numériquement le contenu des features. Cette étape s'appelle *la description*. Un exemple de descripteur peut être le périmètre issu d'une feature représentant un contour. Le descripteur est amené à être comparé à des seuils permettant la dissociation des classes. *Un vecteur de caractéristiques* est un vecteur regroupant plusieurs descripteurs. Un bon descripteur doit avoir une faible variance intra-classe et une forte variance inter-classe. Travailler avec des vecteurs de caractéristiques permet alors de classifier plus de classes ou des classes difficilement dissociables si l'on vient à n'utiliser qu'un seul descripteur. La figure 2.2 permet de visualiser le rapport qui existe entre toutes les notions qui viennent d'être citées.

2.1.1.2 Les features

Dans la mesure où les features sont la base de la détection d'objets (ou de scène), il semble important d'avoir une bonne idée de ce que l'on retrouve derrière ce mot. Comme dit précédemment, les zones d'intérêts sont un sous-groupe de l'ensemble des features. Ils sont le résultat d'un premier traitement de l'image et permettent par la suite la révélation de nouvelles features. On retrouve principalement les contours et les points d'intérêts.

— **Les contours** des objets ou des scènes dans une image sont généralement des régions à forte variation de contraste. Ils permettent d'avoir une information sur la forme de l'objet et de le limiter dans l'espace. Les méthodes les plus communes de détection de contours sont données par (MARR et HILDRETH, 1980), (CANNY, 1986) et (BALLARD, 1981). Si les objets possèdent des formes géométriques telles que des droites ou des

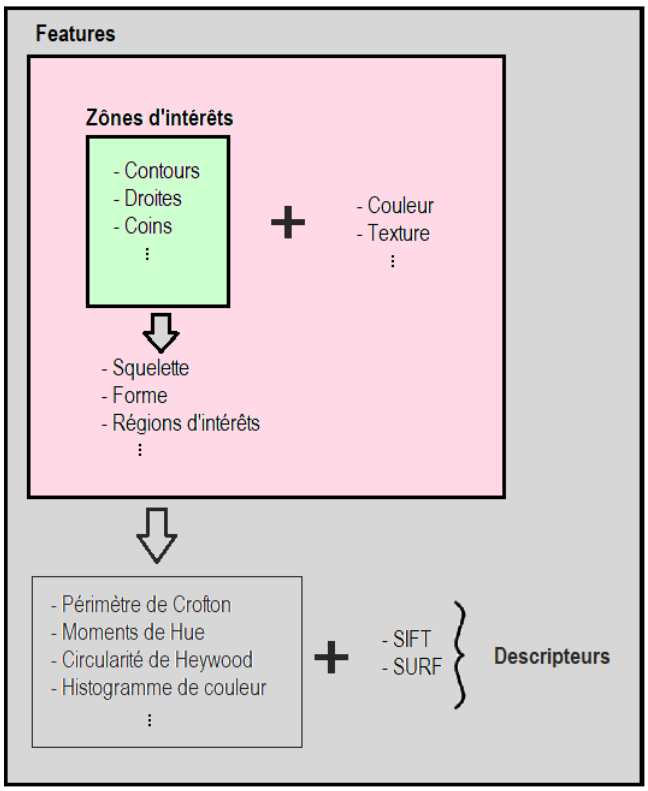

FIGURE 2.2 – Def

cercles, des méthodes comme (ILLINGWORTH et KITTLER, 1988) peuvent être plus adaptées.

— **Les points d'intérêts** sont des points particuliers que l'on retrouve généralement sur des contours. Ils peuvent être des extrémités de droites ou des intersections de courbes. Une des principales méthodes permettant la détection de tels points est donnée par Chris Harris (HARRIS, STEPHENS et al., 1988). L'avantage des détecteurs de points d'intérêts tels que ceux de Chris Harris, c'est qu'ils sont invariants par rotation, par changement d'échelle et par changement de luminosité. D'autres méthodes comme (SHI et al., 1994) et (ROSTEN et DRUMMOND, 2006) permettent la détection de coins dans une image. Ce sont des régions de l'image pour lesquelles il y a deux directions de contours différentes au même endroit.

Ces contours sont utiles à la segmentation d'images 2.1.2.1 et à la création de nouvelles features telles que le squelette morphologique (BLUM et al., 1967). Ce squelette est représenté par un ensemble de courbes centrées dans la forme initiale de l'objet. La taille des branches du squelette ou le nombre de branches sont autant de critères utilisés par la suite à la création de descripteurs utiles à discriminer plusieurs objets entre eux. Si le squelette est un exemple de feature issu de la détection de contour ; les régions d'intérêt en sont l'équivalent pour les points d'intérêt. Ces régions sont des zones centrées autour des points d'intérêt (Figure 2.3). La détection d'un point d'intérêt permet de trouver un endroit de l'image riche en informations. La région

d'intérêt comprend alors l'ensemble de ces informations nécessaires à la comparaison de plusieurs objets. La méthode MSER (Maximally stable extremal regions) par (MATAS et al., 2004) permet de définir de telles régions. Les descripteurs les plus communs issus de telles features sont des histogrammes d'orientation de gradients. Parmi les features non issues des zones d'intérêts, on retrouve principalement la couleur. C'est sûrement la feature la plus largement utilisée dans la mesure où l'on ne s'intéresse qu'à l'intensité des pixels et non pas à leur répartition spatiale. Puisque dans la pratique un objet change rarement de couleur au cours du temps (une voiture, un animal, une veste ...), c'est une excellente feature pour le suivi d'objets. En imagerie une couleur est souvent représentée par ses trois composantes : Rouge, Verte et Bleue. Mais de nombreuses transformations de cet espace couleur existent et permettent ainsi de créer de meilleurs descripteurs selon le contexte étudié. On retrouve notamment l'espace TSL (Teinte, Saturation, Luminosité) ou bien l'espace YUV (une composante de luminance et deux composantes de chrominance) (CHENG et al., 2001). Un dernier type de feature incontournable en détection d'objets est la texture. L'analyse des textures étudie la répartition spatiale de l'intensité des pixels. On retrouve des *textures aléatoires* comme par exemple pour un sol recouvert de paille et des *textures périodiques* pour un sol recouvert de pavés. En 2013 (ABDELMOUNAIME et DONG-CHEN, 2013) présentait une nouvelle base de données pour l'analyse des textures. Les descripteurs sont déterminés soit par une analyse statistique soit par analyse structurelle. Tout ceci n'est qu'une liste non exhaustive des features existantes. Le but est simplement d'avoir une idée de ce qu'est une feature, de comment les extraire et de la façon dont on les utilise.

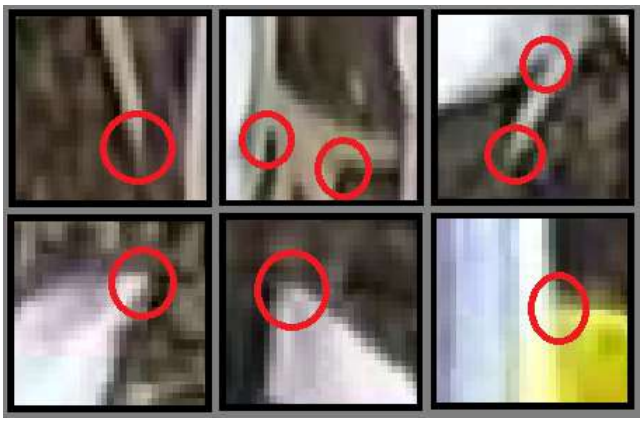

FIGURE 2.3 – Exemple de régions d'intérêts

2.1.1.3 Les descripteurs

Un descripteur est une représentation numérique de ce que traduit une feature. Un feature peut générer plusieurs descripteurs. Par exemple, connaissant le contour d'un objet, on peut en déterminer l'aire, le périmètre, la longueur de l'axe médian, le nombre de branches du squelette, etc. On reconnaît

deux sortes de descripteurs : les descripteurs globaux et les descripteurs locaux.

Les descripteurs globaux sont calculés sur l'ensemble de l'imagette, ou de l'image s'il n'y a pas eu de segmentation au préalable et qu'il n'y a qu'un seul objet à classer. Les features utilisées pour mesurer ces descripteurs sont, eux aussi, globaux (tout sauf les régions d'intérêts). Voici une liste non exhaustive des différents descripteurs globaux que l'on peut retrouver.

— L'aire, le périmètre ou le diamètre de l'objet.
— La direction par intercepts (LAUNEAU et ROBIN, 1996) mesure $N(\alpha)$ qui correspond au nombre de points d'intersection entre le contour et un certain nombre de droites coupant l'image selon la direction α. Figure 2.4, image A.
— Le diamètre de Feret $D(\alpha)$ qui est la plus grande distance entre deux points de contours selon l'axe α. Figure 2.4, image B.
— La compacité, qui est un rapport entre le carré du périmètre et l'aire.
— Les moments d'une image. Ce sont des moyennes pondérées des pixels d'une image, dont la pondération est choisie de sorte à retrouver quelques propriétés intéressantes. Ces propriétés tournent autour de l'intensité totale, du centre de gravité et de l'orientation de l'objet. On définit ces moments comme :

$$\mu_{ij} = \sum_{y=0}^{L-1} \sum_{x=0}^{C-1} (x - \bar{x})^i . (y - \bar{y})^j . I(x, y)$$

avec L et C le nombre de lignes et de colonnes, et $I(x, y)$ l'intensité du pixel en (x, y). Par exemple, la dispersion autour du centre de gravité se mesure par μ_{02} et μ_{20}. Les moments de Hue (HU, 1962) sont de combinaisons non linéaires des moments précédemment cités. Ils sont invariants par rotation, changement d'échelle et translation.
— Une texture peut être vue comme motif basique de quelques pixels qui est répété sur l'ensemble de l'objet. Une étude faite par Tuceryan (TUCERYAN et JAIN, 1993) reprend les différents descripteurs de texture existants.
  — Le **M**otif **B**inaire **L**ocal (MBL) est un exemple de descripteur de texture basé sur une mesure de distance. Pour un pixel donné, ce descripteur accorde une valeur en fonction du signe de la différence avec chacun des pixels voisins et pondéré selon la position du pixel voisin. Il est défini comme :

$$MBL(x_c, y_c) = \sum_{p=0}^{P-1} 2^p \delta(g_p - g_c)$$

P correspond au nombre de voisins étudiés. δ est la fonction de Heaviside. g_c et g_p correspondent respectivement à l'intensité du pixel en c

et du pixel en p. Il est alors possible de créer une distribution des différentes valeurs de $MBL(x)$ (Motif Binaire Local) des pixels qui composent l'objet.

— Une deuxième famille correspond aux descripteurs de texture basés sur des matrices de cooccurrence (EICHKITZ, AMTMANN et SCHREILECHNER, 2013), (PARTIO et al., 2002). La texture est étudiée à partir de la mesure d'occurrence qui existe entre deux pixels d'intensité u et v. Cette matrice est définie comme suit :

$$C_{d_x,d_y}(u,v) = \sum_{x=1}^{n}\sum_{y=1}^{m}\begin{pmatrix}1, si\ \ I(x,y) = u\ \ \&\ \ I(x+d_x, y+d_y) = v\\ 0, sinon\end{pmatrix}$$

Une image I avec p valeurs de pixels différentes, produit une matrice de taille pxp.
Exemple :

$$I = \begin{pmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 1 & 2\\ 0 & 2 & 1 & 2\\ 2 & 1 & 0 & 1\end{pmatrix} \qquad C_{1,0} = \begin{pmatrix}2 & 4 & 1\\ 4 & 4 & 4\\ 1 & 4 & 0\end{pmatrix}$$

À partir de cette matrice, il est possible de mesurer jusqu'à 14 attributs qui permettent de caractériser la texture de l'imagette (homogénéité, grossièreté, périodicité, etc.) (HARALICK, SHANMUGAM et DINSTEIN, 1973).

— La transformée de Fourier (ZHOU, FENG et SHI, 2001) est également un excellent outil pour traduire les effets de périodicité que l'on retrouve dans les textures grâce à l'analyse des fréquences présentes dans les images.

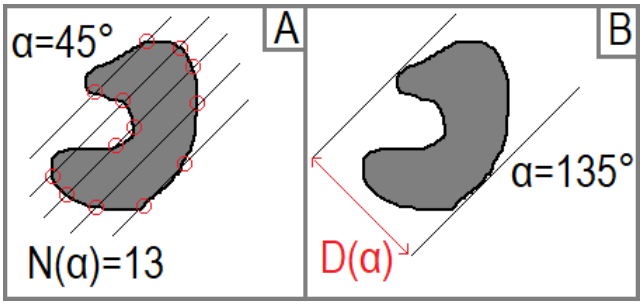

FIGURE 2.4 – Caractéristiques spatiales de descripteurs globaux A : Direction par intercepts, B : Diamètre de Feret

Les descripteurs locaux. Ces descripteurs sont mesurés à partir des features basés sur des régions d'intérêts. Ils décrivent la région autour du point d'intérêt et se présentent généralement sous la forme d'un vecteur. Les descripteurs SIFT (**S**cale **I**nvarient **F**eatures **T**ransform) font partie des références dans le domaine (LOWE, 1999). Un descripteur SIFT est mesuré en prenant une région de 16 pixels par 16 pixels autour du point d'intérêt. Cette zone est ensuite divisée en 4 fois 4 blocs pour lesquels on calcule la norme et l'orientation du gradient enregistré pour 8 directions différentes. Ces valeurs sont ensuite normalisées et concaténées pour former un vecteur de $16 * 8 = 128$ valeurs. Dans la même catégorie on retrouve les descripteurs SURF (Speeded up robust features) (BAY, TUYTELAARS et VAN GOOL, 2006). Ils décomposent, eux aussi, la région d'intérêt en sous-blocs mais contrairement aux SIFT, les SURF mesurent la décomposition en ondelettes de Haar (HAAR, 1910). Les descripteurs GLOH (Gradient Location and Orientation Histogram) (MIKOLAJCZYK et SCHMID, 2005) fonctionnent comme les descripteurs SIFT mais sur des régions d'intérêts plus larges. Le choix du descripteur est donc lié à la résolution de l'objet dans l'image.

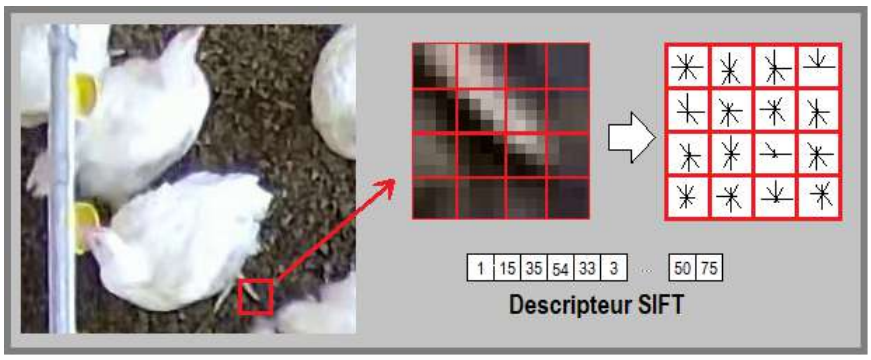

FIGURE 2.5 – Exemple d'un descripteur SIFT

Dans la mesure où les trois étapes, *segmentation, classification, localisation* sont toutes dépendantes de l'extraction de features, il n'est pas rare de faire face à des modèles regroupant la segmentation, la classification et la localisation en seulement une ou deux opérations. Par exemple, la segmentation sémantique correspond à une segmentation plus une classification. Chaque pixel de l'image est alors associé à une classe (GARCIA-GARCIA et al., 2018). La segmentation d'instance est quant à elle un regroupement de ces trois opérations. Chaque pixel est associé à la fois à une classe et à un objet bien précis (HAFIZ et BHAT, 2020). Mais afin d'avoir une meilleure compréhension de ce qu'apporte chaque étape, nous les présentons ici chacune individuellement.

2.1.1.4 Comparaison de méthodes

Aujourd'hui, les deux grandes familles de détection sont d'une part les modèles d'apprentissage profond et d'autre part les méthodes traditionnelles de traitement d'images. La capacité de détection des modèles d'apprentissage profond (basés sur des réseaux de neurones) dépasse de très loin nombre de

méthodes traditionnelles de vision par ordinateur. O'Mahony (O'MAHONY et al., 2019) compare justement les avantages et inconvénients de l'apprentissage profond par rapport aux méthodes traditionnelles de traitement d'images. Globalement, les méthodes d'apprentissage profond obtiennent de bien meilleurs résultats dans les domaines de la classification, de la segmentation et de la localisation d'objets. Ce sont également ces méthodes qui sont capables de regrouper ces trois opérations en une seule étape. D'un autre côté, le fonctionnement de ce genre de méthodes nécessite une phase d'apprentissage des features. La génération d'une grosse base de données comme (LIN et al., 2014) et les moyens de calculs nécessaires à un tel apprentissage (MITTAL, 2019), peuvent être un frein à l'utilisation de ce type de méthodes. Les méthodes traditionnelles de vision par ordinateur bénéficient quant à elles d'une plus grande simplicité algorithmique. Elles sont plus facilement transposables d'un contexte à un autre, là où les méthodes par apprentissage sont limitées aux informations comprises dans leur base de données. En revanche, le choix des features à utiliser pour la détection d'objets n'est pas automatique. Selon le contexte, ces features doivent être robustes à la variation de luminosité, aux occlusions, aux rotations de la caméra ou à la densité des objets dans l'image etc.. Cette demande de robustesse s'accompagne généralement du réglage de nombreux paramètres seuils qui ne sont pas automatiques à fixer. Lorsque la base de données qui permet d'entraîner le modèle de réseau de neurones représente suffisamment bien l'ensemble des situations, alors ce réseau de neurones une fois entraîné devient beaucoup moins sensible aux variations extérieures telles que les changements de luminosité par exemple.

2.1.2 Méthodes classiques de traitement d'images

Nous présentons ici les trois étapes de détections d'objets qui sont : la segmentation, la classification et la localisation.

2.1.2.1 La segmentation

On partitionne les méthodes de segmentation en trois approches qui sont : l'approche contours, l'approche région et l'approche dynamique.

Approche contours La première approche se base sur les contours des objets. Une telle approche part du principe que la frontière entre deux objets adjacents est facilement détectable. Les premières méthodes utilisent des opérateurs différentiels afin de détecter les fortes variations de niveaux de gris. Le choix de ces opérateurs dépend de la précision désirée, du temps de calcul et du rapport signal sur bruit de l'image. Un opérateur bien connu est le filtre de Canny (CANNY, 1986). La détection de contours par Canny commence par appliquer un filtre Gaussien à l'image afin d'atténuer le bruit. Deux opérateurs de convolution $G_x = [-1\ 0\ 1]^T$ et $G_y = [-1\ 0\ 1]$ sont ensuite appliqués à l'image afin de mesurer la valeur du gradient en chaque point de

l'image selon l'axe vertical et l'axe horizontal. On peut donc avoir une information sur l'amplitude et la direction du gradient en chaque point de l'image. Des opérations supplémentaires peuvent alors être apportées afin de garantir l'appartenance d'un point à un contour. Un résumé des opérateurs de détection de contours est à retrouver dans (SPONTÓN et CARDELINO, 2015).
La seconde famille de méthode est basée sur des modèles de contour actif (KASS, WITKIN et TERZOPOULOS, 1988). Ceux-ci sont formés d'une suite de points mobiles le long d'une courbe. Une position initiale de la courbe est fixée autour de l'objet. Enfin plusieurs équations décrivent l'évolution de cette courbe qui finit par épouser le contour de l'objet étudié.

Approche régions La deuxième approche partitionne l'image en **régions** homogènes. Cette approche se base sur des critères tels que l'intensité, la couleur ou la texture, afin de discriminer les régions entre elles. La méthode Mean Shift Clustering (CHENG, 1995) permet d'agréger des pixels entre eux afin de former des sous-régions de l'image. Cette agrégation se fait autour des maximums d'intensité dans l'espace des critères choisis. Pour chaque point P, on cherche l'ensemble des points présents dans un certain rayon. On mesure alors le centre de gravité des points sélectionnés, et on réitère l'étape précédente autour de ce nouveau point jusqu'à convergence. Le point vers lequel converge l'algorithme est un maximum de densité. Il y a alors autant de maximums d'intensité qu'il y a de régions dans l'image segmentée finale. À l'inverse, une méthode comme celle des coupes (théorie des graphes) permet de diviser l'image initiale en différentes régions. Une méthode telle que (FELZENSZWALB et HUTTENLOCHER, 2004) mesure la dissimilarité notée S qu'il peut y avoir entre deux pixels i et j dont les k critères (couleurs, textures, ...) sont notés f_i et f_j.

$$S(f_i, f_j) = \sqrt{\left(\sum_k (f_{ik} - f_{jk})^2\right)}$$

La segmentation de l'image résulte alors d'une minimisation des coûts globaux liés aux indices de dissimilarité de chaque région et de façon proportionnelle à leur taille.

La figure 2.6 illustre un cas de segmentation par analyse de contours et un cas de segmentation par analyse de régions. À noter que des étapes de débruitage sont souvent nécessaires afin d'être moins sensible aux fortes variations locales d'intensités.

Approche dynamique Approche par **soustraction de fond**. Cette méthode permet de détecter des objets par différence d'image avec une image de fond. On a donc besoin d'une image de référence dépourvue d'objets à détecter (image B 2.1). La caméra doit être fixe afin d'assurer une image de référence identique à chaque instant de la vidéo. Lorsqu'un objet apparaît sur l'image (image A 2.1), la différence entre cette image et l'image de référence fait alors

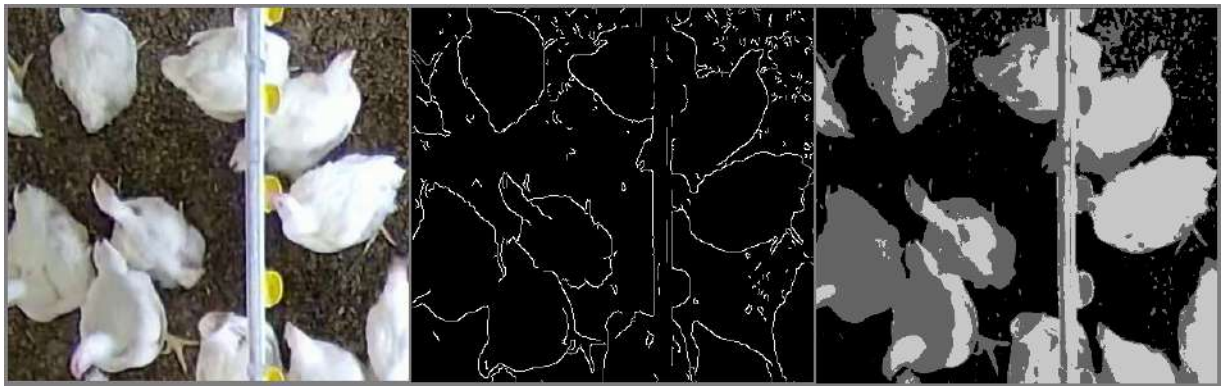

FIGURE 2.6 – A gauche : Image initiale, Au centre : segmentation contours, A droite : Segmentation région

ressortir tout objet non présent sur l'image de référence (image C 2.1). Enfin chaque tache peut alors être considérée comme un objet à localiser (image D 2.1).

2.1.2.2 La classification

La classification permet d'affecter un nom à un type d'objet ou de scène bien précis, et ce, de façon automatique. Si l'on se réfère à l'image segmentée B de la Figure 2.1, on retrouve cinq imagettes à classifier. Les différents noms de classes qui peuvent être affectés à ces imagettes sont : *Poulet*, *Mangeoire* et *Arrière-plan*. Sur l'image C de cette même figure, chaque imagette s'est vue affecter le nom d'une classe. C'est le résultat d'une classification d'images. Les descripteurs ne sont que des intermédiaires dans le processus de classification. Car, à moins d'avoir des imagettes identiques, il est alors peu probable d'avoir exactement les mêmes valeurs de descripteurs pour chaque classe. Les classes sont finalement représentées par plusieurs descripteurs, mais dont les proportions varient d'une classe à l'autre. L'ensemble de ces descripteurs est regroupé dans ce que l'on appelle : des vecteurs de caractéristiques. Ces vecteurs peuvent être représentés dans un espace ayant autant de dimensions qu'il y a de descripteurs. C'est l'espace des descripteurs (Figure 2.8). La recherche de clusters dans cet espace des descripteurs permet de définir les différentes classes. À présent on doit faire face à deux cas de figure. Soit les classes sont connues d'avance, c'est ce que l'on appelle : la classification supervisée, soit les classes sont inconnues et l'on fait de la classification non supervisée. La revue de D. Lu reprend l'ensemble des méthodes et techniques pour améliorer les performances des classifications d'image. (LU et WENG, 2007).
On définit les notations suivantes :
— $I_1, \ldots, I_n$ sont les imagettes à classifier
— $x_1, \ldots, x_n$ sont les vecteurs de caractéristiques
— $x_k[i]$ est le i^{eme} descripteur du k^{eme} vecteur de caractéristiques
— $C_1, \ldots, C_k$ sont les vecteurs de caractéristiques de référence des k classes dans l'espace des descripteurs

Afin d'être plus précis dans notre vocabulaire, on distingue deux types de classes. Il existe d'une part *les classes d'information* et d'autre part *les classes spectrales*. Les classes d'informations sont ces catégories d'objets ou de scènes sur lesquels on veut justement mettre un nom (ex : Poulet, Mangeoire, Perchoir, Gamelle, ...). Les classes spectrales sont quant à elles des ensembles de pixels formant les imagettes à l'intérieur desquelles il existe une cohérence des intensités de pixels. Le but est alors de faire correspondre chaque classe spectrale avec une classe d'information. Cette mise en correspondance n'est pas forcément triviale. Par exemple, il peut y avoir plusieurs classes spectrales qui se rapportent à une même classe d'information. L'arrière-plan d'un élevage de poulets peut être composé de paille, de copeau ou de sciure. Et à moins d'avoir affaire à deux imagettes identiques, deux classes spectrales appartenant à une même classe d'informations n'ont que peu de chances d'être également identiques ; d'où l'intérêt de passer par des vecteurs de caractéristiques (Figure 2.7).

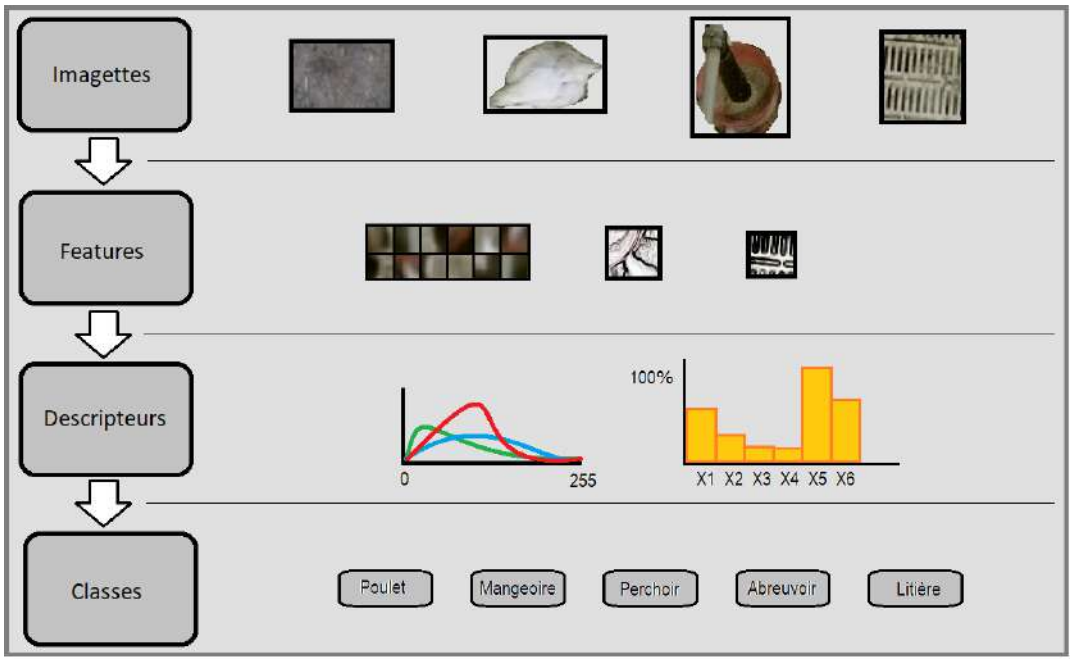

FIGURE 2.7 – schema

La classification supervisée Les méthodes supervisées nécessitent une base de données d'imagettes de référence. Les classes spectrales des imagettes de cette base de données doivent être représentatives des classes d'informations étudiées. Par exemple si l'on cherche à classifier de la paille, de la sciure ou des copeaux, les imagettes ne doivent pas être polluées par n'importe quel objet dont l'information spectrale viendrait biaiser les informations incluses dans les vecteurs de caractéristiques. Ces méthodes cherchent à définir des vecteurs de caractéristiques de référence C_i représentatifs des classes à étudier. Il est souvent nécessaire d'utiliser plusieurs imagettes par classe afin

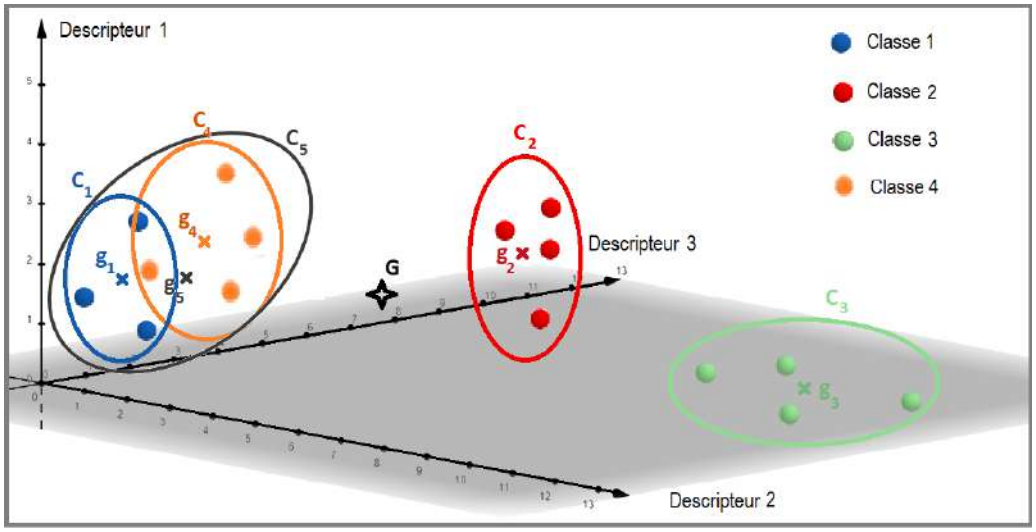

FIGURE 2.8 – Regroupement des classes dans l'espace des features

d'être plus représentatif des classes étudiées. Ces méthodes sont donc composées de deux étapes : la création des vecteurs de caractéristiques de référence (apprentissage) puis de la classification à proprement parler. Comparer deux imagettes (une de référence et une étudiée par exemple), revient à mesurer une distance entre leurs vecteurs de caractéristiques. Plus la distance est faible, plus la probabilité que les deux imagettes appartiennent à la même classe est grande. Cette opération passe par la définition d'une métrique et par une normalisation des descripteurs afin qu'il n'y en ait pas un qui prenne le pas sur les autres.

K - plus proches voisins Les classes de cette méthode sont représentées par de nombreux vecteurs de caractéristiques de référence. Pour un vecteur de caractéristiques donné, correspondant à l'imagette à classifier, on regarde à quelles classes appartiennent les **K** plus proches voisins dans l'espace des descripteurs (FIX et HODGES, 1989). Pour cela, une métrique doit être définie au préalable (ex : distance euclidienne). L'imagette se voit enfin affecter la classe majoritairement présente parmi les **K** voisins. Le choix de la valeur **K** dépend du nombre de dimensions et du nombre de classes (NIGSCH et al., 2006).

Arbres de décision Un arbre de décision permet de répartir les imagettes en sous-classes homogènes en fonction des descripteurs. Cet arbre est composé de branches et de nœuds (ou feuilles). Chaque nœud porte sur un descripteur. Selon la décision établie sur chaque nœud, un certain nombre de branches sont alors formées. Par exemple, si le nœud porte sur un choix binaire (ex : $x[i]0.5$), deux branches sont donc formées à partir de ce nœud. Passent par la première branche tous les x_k dont $x_k[i] > 0.5$ et par la deuxième ceux dont $x_k[i] <= 0.5$. Chaque branche est alors connectée à un nouveau nœud portant sur un autre descripteur. Les dernières branches sont composées de x_k

appartenant tous à une même classe. La phase d'apprentissage sélectionne les descripteurs les plus pertinents à chaque nœud et la décision à prendre pour chaque branche.

Forêt aléatoire Pour la méthode des forêts aléatoires, la base de données d'apprentissage est découpée en sous-catégories. Chaque sous-catégorie permet d'entraîner un arbre de décision différent. Puis chaque échantillon x_k passe par chaque arbre de décision. Le résultat le plus fréquent est retenu comme classe à affecter à l'imagette I_k. La méthode CART pour Classification And Regression Trees de Breiman et al. (BREIMAN et al., 2017) est une méthode de référence permettant de fixer les nœuds des arbres de décision.

À part ça Ces algorithmes n'étant pas les seuls dans le domaine de la classification supervisée, voici une liste non exhaustive d'algorithmes bien connus :
— K plus proches voisins (KIM[1], KIM et SAVARESE, 2012)
— Arbres de décision
— Forêts aléatoires
— Analyse (factorielle) discriminante (COHEN, WEST et AIKEN, 2014)
— Régression logistique (NELDER et WEDDERBURN, 1972)
— Machine à vecteurs de support (SUTHAHARAN, 2016), (KIM[1], KIM et SAVARESE, 2012)

La classification non supervisée Contrairement aux méthodes supervisées, il n'est à présent plus possible de comparer les vecteurs x_k avec des vecteurs de référence dont on connaît la classe. Par contre, ces méthodes permettent de regrouper tous les vecteurs en sous-groupes homogènes. En d'autres mots : si deux vecteurs x_1 et x_2 sont *proches* ils sont alors considérés comme appartenant à la même classe. En revanche s'ils sont *éloignés*, les deux vecteurs correspondent à des classes différentes. Pour ce faire, il faut un moyen de mesurer la similarité qui existe entre eux (BLOOM, 1981). Dans l'espace des descripteurs on note :
— N le nombre de vecteurs de caractéristiques (nombre d'individus)
— x_i les vecteurs de caractéristiques $i \leq N$
— G le centre de gravité de tous les individus
— K le nombre de classes
— C_i la i^{eme} classe avec $i \leq K$
— P_i le nombre d'individus appartenant à la classe C_i
— $g_i \in \mathcal{R}^p$ le centre de gravité des individus appartenant à la classe C_i et p le nombre de descripteurs
— $\mathcal{C}(t) = \left\{g_1^{(t)}, g_2^{(t)}, \ldots, g_2^{(t)}\right\}$ est l'ensemble des centres des classes à l'instant t
— J_W l'inertie intra-classes

$$J_W = \sum_{j=1}^{K} \sum_{x \in C_j} d(x, g_j)$$

— J_B l'inertie inter-classes

$$J_B = \sum_{i=1}^{K} P_i d(G, g_i)$$

— J_{tot} l'inertie totale J_{tot}
On recherche donc à regrouper les vecteurs en groupes homogènes de telle sorte que l'inertie inter-classe soit la plus grande possible et l'inertie intra-classe la plus faible possible, pour un nombre K de classes fixé. L'énergie totale est indépendante du nombre de classes et reste constante. Diminuer l'inertie intra-classe revient automatiquement à augmenter l'inertie inter-classes. Il existe pour un nombre K de classes fixé, une répartition des individus en K groupe qui retourne une inertie inter-classes maximale. La différence entre les différentes méthodes de classification non supervisée se joue d'une part sur la connaissance a priori ou non du nombre K de classes, et d'autre part sur la mesure de similarité entre deux individus.

Mesure de similarité

Comme dit précédemment, on cherche à mesurer une similitude entre les imagettes d'une même classe spectrale. Cela passe par une mesure de similitude entre les vecteurs de caractéristiques. Un indice de similarité est une valeur comprise entre 0 et 1, où une valeur de zéro traduit une totale différence entre deux vecteurs et une valeur de un représente deux vecteurs égaux. Le complément d'un indice de similarité donne la mesure de la dissimilarité, $dissimilarite = 1 - similarite$. Un indice de dissimilarité est une application de ExE dans $\mathcal{R}^+$ avec E l'ensemble des N vecteurs de caractéristiques à classer. Un indice de dissimilarité remplit les conditions suivantes :
— d(i,j) = d(j,i) 0
— d(i,i) = 0
L'inégalité triangulaire n'est pas une condition nécessaire pour un indice de dissimilarité. Une distance satisfait ces conditions. Une mesure de distance comme la distance Euclidienne traduit une dissimilarité dont la valeur va jusqu'à $+\infty$. Dans ce cas, il n'existe pas de complément de similarité. Il peut être intéressant de centrer et réduire les valeurs des descripteurs afin d'éviter que certains d'entre eux ne prennent l'ascendant sur les autres lors de la mesure de dissimilarité. Sans cela, la dissimilarité entre deux vecteurs ne dépendrait que de la dissimilarité entre deux descripteurs particuliers. Voici une liste non exhaustive des différents indices et distances que l'on peut utiliser :
— Distance de Camberra (LANCE et WILLIAMS, 1966)

$$D(u, v) = \sum_{i=1}^{p} \frac{|u_i - v_i|}{|u_i| + |v_i|}$$

— Coefficient de corrélation de Pearson (BENESTY et al., 2009)

$$r(u,v) = \frac{\sum_{i=1}^{p}(u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^{p}(u_i - \bar{u})^2}\sqrt{\sum_{i=1}^{p}(v_i - \bar{v})^2}}$$

— L'indice de Jaccard (NIWATTANAKUL et al., 2013)

$$J(u,v) = \frac{u \cap v}{u \cup v}$$

— Distance de Manhattan

$$D(u,v) = \sum_{i=1}^{p} |u_i - v_i|$$

In fine, les méthodes non supervisées permettent donc de recouper des imagettes qui possèdent les mêmes descripteurs sous un même label, sans savoir à quel type d'objet correspond ce label. Parmi ces méthodes on retrouve notamment : l'analyse par composante principale (WOLD, ESBENSEN et GELADI, 1987), la décomposition en valeur singulière (SAVAS et ELDÉN, 2007), la classification hiérarchique (MURTAGH et CONTRERAS, 2011), la classification par moyennes mobiles (MACQUEEN et al., 1967), l'algorithme espérance-maximisation (DEMPSTER, LAIRD et RUBIN, 1977). Nous présentons ici une méthode par partitionnement (k-means) et une méthode de regroupement hiérarchique (Classification hiérarchique ascendante).

K-means

Pour la méthode des K-means (ou moyennes mobiles), soit le nombre K de classes est connu a priori, soit plusieurs tests sont effectués en faisant varier la valeur de K afin de trouver la valeur produisant les meilleures inerties. Les étapes sont les suivantes :

Algorithm 1 méthode des K-means

Require: Fixer le nombre K de classes
Ensure: S'assurer que $N \geq K$
  Initialisation aléatoire de $\mathcal{C}(0)$ ▷ l'ensemble des centres des classes
  while $\mathcal{C}(t) \neq \mathcal{C}(t-1)$ **do**
    Chaque x_i est assigné à la classe dont le centre g_j est le plus proche
    Redéfinition de $\mathcal{C}$ en fonction des assignations précédentes
  end while

Classification hiérarchique ascendante

Cette méthode de classification permet de recouper petit à petit chaque individu en sous-groupes de plus en plus importants jusqu'à ce que le nombre de classes souhaité soit atteint.

— Mesure la dissimilarité entre les N individus x_i

— On forme un sous-groupe avec les deux individus dont la mesure de similarité est la plus forte (ou la distance la plus faible)
— Ce sous-groupe correspond donc à une nouvelle classe C_1 centrée en g_0
— Puis on recommence l'opération avec les N-2 individus restants et le nouveau point g_0
— Ainsi, on regroupe à chaque itération soit deux individus entre eux, soit un individu avec une classe.
— L'algorithme s'arrête lorsque le nombre de classes souhaité est atteint.

Recoupement de classes Le recoupement de classes est utilisé : soit pour des méthodes telles que la classification hiérarchique ascendante, qui visent à diminuer le nombre de classes de façon optimale, soit lorsque le nombre de classes n'est pas connu a priori. Nous avons dit précédemment que pour un nombre K de classes fixé, l'objectif est de diminuer l'inertie intra-classes. Il se trouve qu'en diminuant le nombre de classes, l'inertie intra-classe optimale augmente. Une méthode de recoupement de classes cherche les classes à fusionner qui minimisent cette augmentation de l'inertie intra-classe. La méthode de Ward (WARD JR, 1963) cherche la fusion de classes qui minimise cette augmentation.

Exemple
Nous prenons ici l'exemple des classes de la figure 2.8. Sur cette figure les classes C_1 et C_4 semblent très proches, et il pourrait être envisagé de les fusionner sous une classe C'. L'équation suivante donne la distance entre cette nouvelle classe et une classe j.

$$D(C', C_j) = \frac{(P_j + P_1)\, D\,(C_1, C_j) + (P_j + P_4)\, D\,(C_4, C_j) - (P_1 + P_4)\, D\,(C_1, C_4)}{P_j + P_1 + P_4}$$

La recherche du nombre de classes optimal n'est pas du tout triviale. La méthode Ward permettant de minimiser la variation de l'énergie intra-classe, une recherche d'un "coude" dans la courbe représentant cette variation en fonction du nombre de classes permet de donner une idée de la valeur optimale de K.

2.1.2.3 La localisation

Une fois l'image segmentée et ses éléments classifiés, vient le moment de localiser plus précisément les objets. Généralement, localiser un objet revient à donner sa position (son centre de gravité) et sa taille. La taille de l'objet peut être donnée en retournant le rectangle circonscrit à l'objet, ou bien, si celui-ci est bien isolé dans l'image, en retournant sa région issue de la segmentation. On appelle cette dernière forme de localisation la segmentation d'instance. La complexité de la localisation d'objet apparaît lorsque deux ou plusieurs objets sont accolés. Auquel cas, soit on peut avoir recours à une segmentation plus

fine notamment si le nombre d'objets par région est connu d'avance (de par la taille de la région par exemple), soit on applique des opérations d'érosion (GIL et KIMMEL, 2002) qui consistent à rogner les régions de chaque objet jusqu'à ce que de nouvelles formes se découpent. Enfin, si les objets ont une forme rigide et que cette forme est connue d'avance, il est alors possible de diviser les différents objets en se basant sur la forme de leur squelette (BLUM et al., 1967).

Morphologie mathématique

L'érosion est un opérateur qui permet de réduire la taille d'une imagette. Soit X un sous-ensemble d'une image E correspondant à la zone image que l'on cherche à éroder et soit B un élément structurant correspondant à la forme de l'érosion. La dilatation morphologique de X par B est donnée par :

$$\mathcal{E}(X) = X \ominus B = \{x | B_x \in X\}$$

2.1.3 Méthodes par apprentissage profond

L'apprentissage profond est un sous-domaine de l'apprentissage automatique (**M**achine **L**earning **ML**), lui-même étant un sous-domaine de l'intelligence artificielle. Le ML regroupe l'ensemble des techniques qui permet aux machines d'améliorer leurs performances en se basant sur l'expérience de leurs résultats passés. Bien que toutes ces méthodes ne soient pas basées sur des réseaux de neurones (ex : méthode AdaBoost (FREUND, SCHAPIRE et ABE, 1999)), ceux-ci tiennent une part importante, notamment en traitement d'images. L'apprentissage profond est un sous-domaine de l'apprentissage automatique qui utilise justement les réseaux de neurones avec des structures composées d'un très grand nombre de couches. Comparées aux autres méthodes de ML, les méthodes d'apprentissage profond nécessitent beaucoup plus de quantité de données d'entraînement, plus de ressources de calcul, plus de temps d'entraînement, mais permet de sortir des types de données plus complexes. Pour l'histoire, le premier modèle mathématique de neurones a été créé en 1943 par McCulloch and Pitts (MCCULLOCH et PITTS, 1943). En 1957, Rosenblatt introduit le perceptron (ROSENBLATT, 1958). C'est le plus simple réseau de neurones qui puisse être et qui introduit la notion d'apprentissage. En 1989, Yann LeCun crée les réseaux de neurones convolutifs (**C**onvolutionnal **N**eural **N**etwork **CNN**) (LECUN et al., 1989). En plus du travail de classification que l'on retrouve dans la plupart des modèles de ML, ces modèles possèdent des couches d'extraction de features (caractéristiques visuelles intrinsèques aux objets à détecter), qui permettent justement d'apprendre les caractéristiques visuelles les plus adaptées à la détection des types d'objets que l'on souhaite détecter. Enfin en 2012, Alex Krizhevsky remporte un challenge de reconnaissance d'objets grâce à une énorme base de données et à l'utilisation de GPUs pour faire tourner son modèle de CNN. À partir de là

les réseaux de neurones convolutifs connaissent un grand essor, qui leur permet d'être aujourd'hui les modèles de prédilection de détection d'objets.

2.1.3.1 Le perceptron

Le perceptron (Figure 2.9) est composé de plusieurs entrées $X1, ..., Xn$ chacun connecté à un neurone lié à un coefficient (ou poids) $Y1, ..., Yn$. La fonction de combinaison qui suit permet de combiner les entrées pondérées de leur poids comme suit :

$$Z = b + \sum_i X_i W_i$$

La fonction d'activation qui suit est l'équivalent du potentiel d'activation des neurones biologiques. Une valeur positive de sortie S1 apparaît si la valeur de Z est suffisamment grande. Afin de pouvoir approximer n'importe quelle fonction, la plupart de ces fonctions d'activation sont non-linéaires. Parmi elles on retrouve : la fonction Sigmoïde, Heaviside, la tangente hyperbolique ou la fonction Unité Linéaire Rectifiée (ReLU) pour les plus populaires.

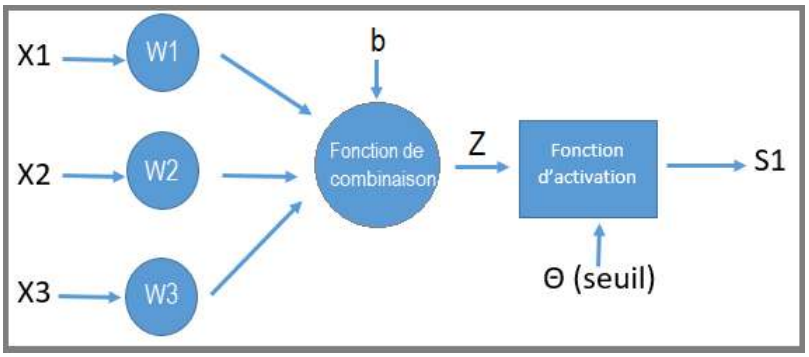

FIGURE 2.9 – A : Le perceptron

Dans le cas des méthodes supervisées, le résultat prédit est comparé avec le résultat connu. Une rétropropagation de la différence est effectuée afin de permettre d'ajuster les poids. Ce perceptron est donc l'élément de base de tout réseau neuronal. De façon générale, les réseaux neuronaux sont composés de plusieurs perceptrons connectés. La figure 2.10 représente ce que l'on appelle un perceptron multicouches. Il est composé d'une couche d'entrée, d'une ou plusieurs couches cachées et d'une couche de sortie. Une architecture qui possède plus de deux couches cachées est généralement considérée comme profonde (BENGIO, 2009). Le perceptron multicouches est un type de réseau de neurones à propagation avant. L'information ne va que de l'avant, des nœuds de la couche d'entrée à ceux de la couche de sortie. Dans les réseaux plus complexes, on trouve des boucles qui renvoient une partie des informations dans les couches précédentes du réseau.

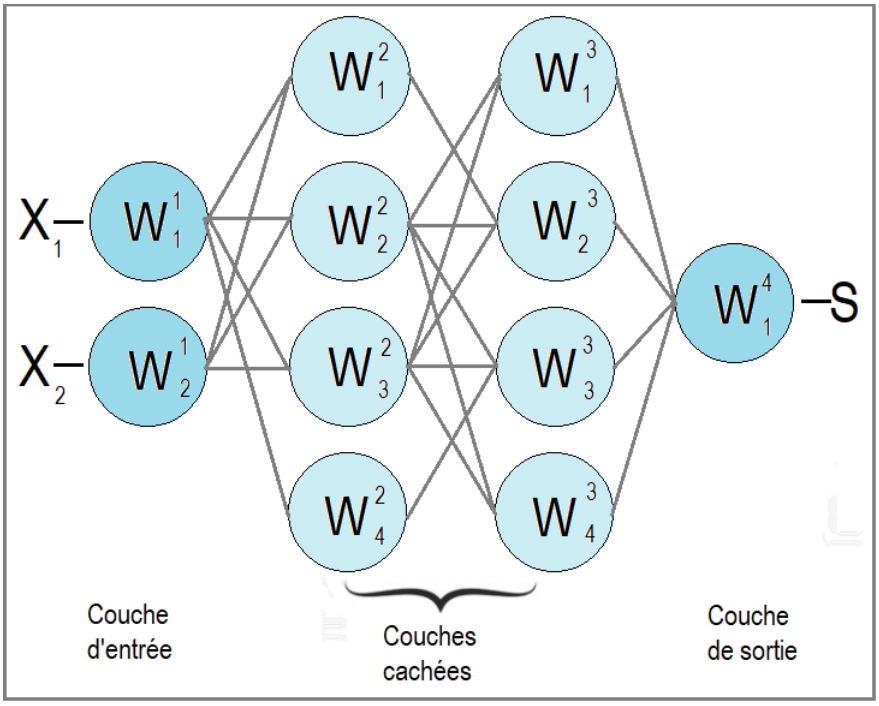

FIGURE 2.10 – A : Un perceptron multi couche

2.1.3.2 Les réseaux de neurones convolutifs

Un réseau de neurones convolutif ou CNN (LECUN et al., 1989), est un modèle qui permet entre autre la détection d'objets dans une image. L'intérêt d'un tel modèle, c'est qu'il permet de déterminer automatiquement les features de l'image les plus pertinentes à la détection des objets souhaités. Là où un algorithme de traitement d'images classique est capable de détecter des features simples, basées généralement sur de la détection de contours. Un modèle CNN est lui,capable de trouver des features bien plus complexes et bien plus pertinentes. Un CNN est composé de trois types de couches qui alternent : les couches de convolutions, les couches d'activations et les couches de pooling.

Les couches de convolution
La convolution d'une image permet des opérations telles que le débruitage d'une image, la détection de contours 2.1.2.1 ou l'inversion de couleurs. L'image I' résultant de la convolution entre une image I et un élément convolutif de taille $(2C+1, 2L+1)$ s'écrit :

$$I'(x,y) = I(x,y) * h(x,y) = \sum_{i=-C}^{C} \sum_{j=-L}^{L} h(i,j)\, I(x-i, y-j)$$

La convolution est appliquée en chaque point de l'image initiale. L'image I' possède donc 2C colonnes de moins et 2L lignes de moins que l'image I. C'est cet élément de convolution noté h dans l'équation précédente qui spécifie le

type de features que détecte le réseau de neurones convolutifs. Cet élément de convolution est appelé *filtre* et est souvent représenté sous la forme d'une petite image. Dans un CNN il en existe des milliers et tous différents. Les valeurs de ces filtres sont déterminées au cours de la période d'apprentissage du réseau de neurones. Ces filtres sont donc appris comme le sont les neurones de n'importe quel réseau. Puisqu'il existe une période d'apprentissage avec un réseau de neurones convolutif, cela veut dire qu'il est entraîné avec des images comprenant les objets que l'on cherche à détecter. Un CNN a donc lors de son apprentissage toutes les informations dont il a besoin pour la mise à jour de de ces filtres correspondant aux features des objets à détecter.
Une image subit plusieurs convolutions. Chaque ensemble de convolution engendre de nouvelles images de taille plus petite. Ces nouvelles images subissent à leur tour un ensemble de convolutions comme représenté sur la Figure 2.11. À noter que les poids du réseau correspondent seulement aux filtres de convolutions qui sont de petites tailles (généralement de 3x3 à 7x7) et non pas aux images créées. Les convolutions des premières couches permettent généralement de détecter des droites. Plus on avance dans les couches plus les convolutions permettent de détecter des features beaucoup plus fines.

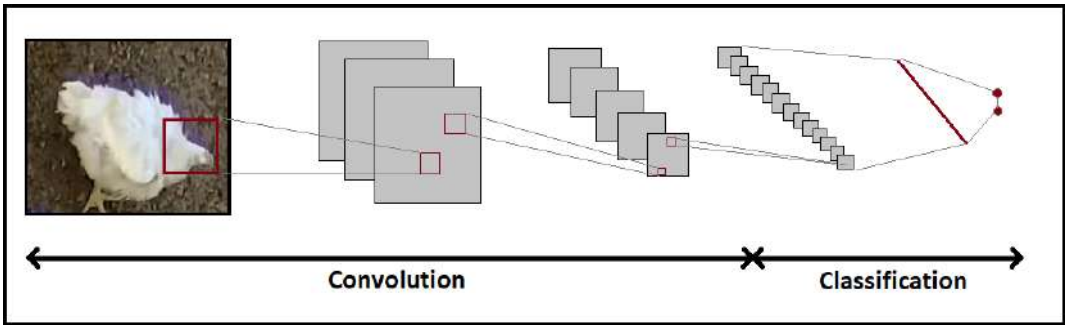

FIGURE 2.11 – Convolution d'un CNN

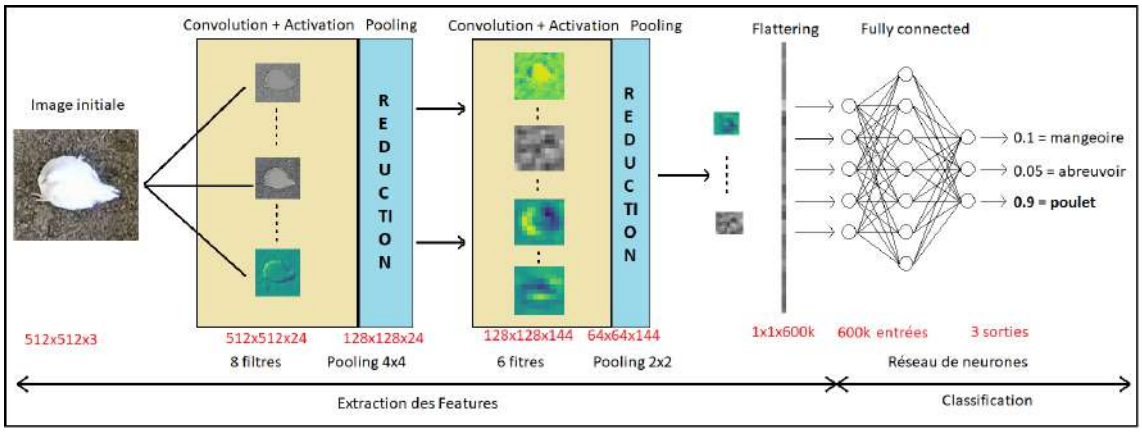

FIGURE 2.12 – Architecture d'un réseau de neurones convolutif

Les couches d'activation

La couche d'activation permet la même opération que l'activation que l'on retrouve à la sortie de chaque neurone d'un perceptron 2.1.3.1. Cela permet comme dit précédemment d'insérer de la non-linéarité dans le modèle. Il existe de nombreuses fonctions d'activation (SHARMA, SHARMA et ATHAIYA, 2017) dont le choix dépend du problème à résoudre. Les propriétés de ces fonctions d'activation sont :

— Les bornes de sortie. Une couche de sortie demande des valeurs entre 0 et 1. Une couche cachée n'a pas forcément de limites.
— Le temps de calcul. Une fonction faisant intervenir un terme exponentiel est plus lente qu'une fonction linéaire.
— La dérivée. Plus une fonction a sa dérivée proche de zéro, moins la rétropropagation sera efficace.
— La fonction nulle. Si la fonction est nulle sur une trop grande plage de valeurs (généralement les valeurs négatives), il existe un risque de *mort* de certains neurones que l'on ne peut plus récupérer. D'un autre côté ce type de fonction permet d'augmenter l'efficacité du temps de calcul.

Parmi les fonctions d'activation les plus connues sont :

— La fonction sigmoïde $sigmoide(x) = \frac{1}{1+\exp^{-x}}$. Elle est principalement utilisée pour de la classification binaire.
— La fonction Rectified Linear Unit (ReLU) $ReLU(x) = max(0, x)$. C'est la fonction la plus utilisée dans les couches intermédiaires.
— La fonction Softmax $softmax(x) = \frac{\exp^x}{\sum_i \exp^{x_i}}$. Cette fonction est surtout utilisée dans les dernières couches d'un réseau de classification multiclasses.
— La fonction SoftPlus $softplus(x) = log\,(\exp^x + 1)$. C'est l'équivalent de la fonction ReLU. Le fait que la fonction soit plus lisse que la fonction ReLU permet une meilleure rétropropagation de mise à jour des poids. En revanche les calculs sont plus lourds à cause des fonctions logarithme et exponentielle.

Les couches de pooling

Le pooling (*mise en commun*) permet de réduire la taille de l'image d'entrée et ainsi de réduire les temps de calcul des couches suivantes. Les couches de convolutions permettent de détecter des features utiles à la détection d'objets. Ces features peuvent être des détails très fins de l'image. Réduire la taille de l'image, permet de travailler sur des features plus grossieres et ainsi d'être moins sensible aux petites variations de l'image d'entrée. L'utilisation de couches de pooling est obligatoire au bon fonctionnement d'un réseau convolutif si celui-ci possède une assez grande base de données pour couvrir toutes les invariances (SPRINGENBERG et al., 2014). Si ce n'est pas le cas le risque de surapprentissage devient important. L'image est découpée sous forme de *tuiles* qui représentent un groupe de n par m pixels. Chaque tuile est traduite en un unique pixel dans la couche de sortie dont la taille est alors réduite d'un facteur nxm 2.13. Parmi les différentes couches de pooling on retrouve :

— Le max pooling. Chaque pixel de l'image de sortie correspond à la valeur maximale de la tuile associée. Il permet de faire ressortir les features fortement marquées comme des droites verticales par exemple.
— Le mean pooling. Chaque pixel de l'image de sortie correspond à la moyenne des valeurs (arrondie) des pixels de la tuile associée. Il a tendance à lisser l'image et permet de faire ressortir des features peu marquées.

Les tuiles utilisées dans les couches de pooling sont généralement de taille 2x2 ou 3x3. Utiliser de petites tailles évite au réseau de perdre trop d'informations d'un coup.

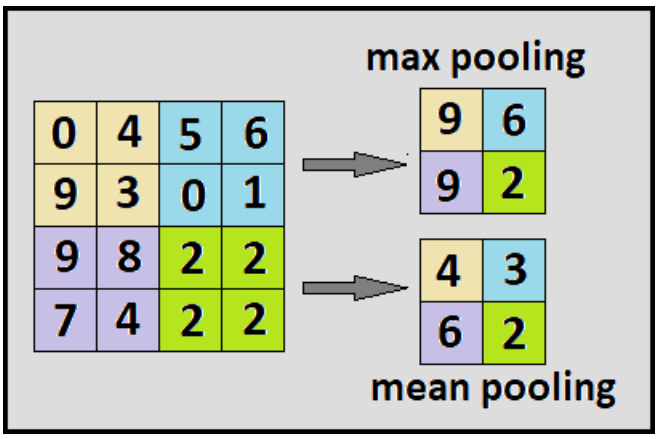

FIGURE 2.13 – Pooling 2x2

Les couches fully connected

Ces couches ne sont pas spécifiques des réseaux convolutifs. Dans le cas des CNNs, elles sont utilisées en toute fin du réseau de neurones. La dernière couche de convolution est transformée en un vecteur, en une dimension ou chaque neurone est connecté à tous les neurones de la dernière couche (fully connected). Cette dernière couche contient autant de neurones qu'il y a d'objets à classer (voire Figure 2.12). À noter qu'une couche d'activation de type sigmoïde est utilisée afin de garantir un résultat pour chaque classe compris entre 0 et 1. Une valeur de zéro traduit une probabilité nulle d'appartenance de l'objet à la classe représentée par ce neurone de sortie. Une valeur de 1 traduit une probabilité de 100% d'appartenance à la classe en question.

2.2 Matériel et captations

2.2.1 Matériel

On veut pouvoir analyser le comportement d'un maximum de poulets dans un élevage. Nous avons à notre disposition plusieurs caméras filaires Bascom ainsi qu'un enregistreur 4 canaux de capacité de stockage de 1 To. On retrouve des caméras de 1 mégapixel (1280 x 720), 2 mégapixels (1920x1080) et des caméras 3 mégapixels (2048x1536). Ces caméras sont certifiées IP66,

c'est-à-dire : anti-poussières, imperméables aux forts jets d'eau venant de n'importe quelle direction. C'est la norme la plus couramment rencontrée pour les caméras de surveillance. Celles-ci possèdent un angle de vision allant jusqu'à 92° et possèdent une mise au point automatique. L'enregistreur possède une programmation qui permet de sélectionner les périodes d'enregistrement sur toute la période du lot d'élevage. En revanche l'enregistrement des vidéos quoiqu' étant continu, crée des morceaux de vidéos de l'ordre de 8 minutes qu'il faut alors recouper entre eux pour plus de facilité de traitement. Travailler avec des caméras à large champ permet d'analyser une plus grande surface au sol avec une seule et même caméra, ce qui est un net avantage lorsque l'on a pour objectif d'analyser le comportement animal dans des élevages de l'ordre de 1500 m². L'inconvénient étant que ce genre de caméras subissent de fortes distorsions optiques.

2.2.2 Distorsions

Les distorsions optiques sont toutes déformations géométriques de l'image engendrées par l'objectif. On retrouve deux types de distorsion : la distorsion radiale et la distorsion tangentielle. La distorsion radiale est causée par la forme sphérique de la lentille. La lumière passant par le centre de la lentille ne subit que très peu de réfraction contrairement à la lumière passant par les extrémités de la lentille. C'est cette zone qui crée le plus de distorsion. Cette distorsion radiale se traduit par la transformation de droites en courbes. On retrouve deux types de déformations radiales qui sont : la distorsion en barillet et la distorsion en coussinet (Figure 2.15). Les distorsions en barillet sont typiques des déformations que l'on retrouve avec des objectifs grands-angles. Deux objets de même taille à égale distance de l'objectif ont alors une taille différente sur l'image. Cette taille est fonction de leur distance à l'axe optique de la caméra (Figure 2.14).Elle est écrite comme :

$$D(\%) = \frac{H - H'}{H} * 100 \tag{2.1}$$

Avec H et H' respectivement les demies diagonales de l'image initiale et de l'image distordue (Figure 2.15) et D la mesure de distorsion.
Dans notre cas cette distorsion est mesurée à : $D(\%) = 13\%$. Par convention, une distorsion supérieure à 1% est considérée comme gênante.
La distorsion tangentielle dépend de la conception du système optique de l'objectif. Elle apparaît lorsque la lentille n'est pas parallèle au plan de l'image. Cela donne un effet trapèze à l'image. Certaines zones de l'image semblent plus proche de l'observateur.
Fort heureusement, il est possible de corriger de telles déformations. La distorsion radiale s'écrit :

$$x_{distorded} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

$$y_{distorded} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

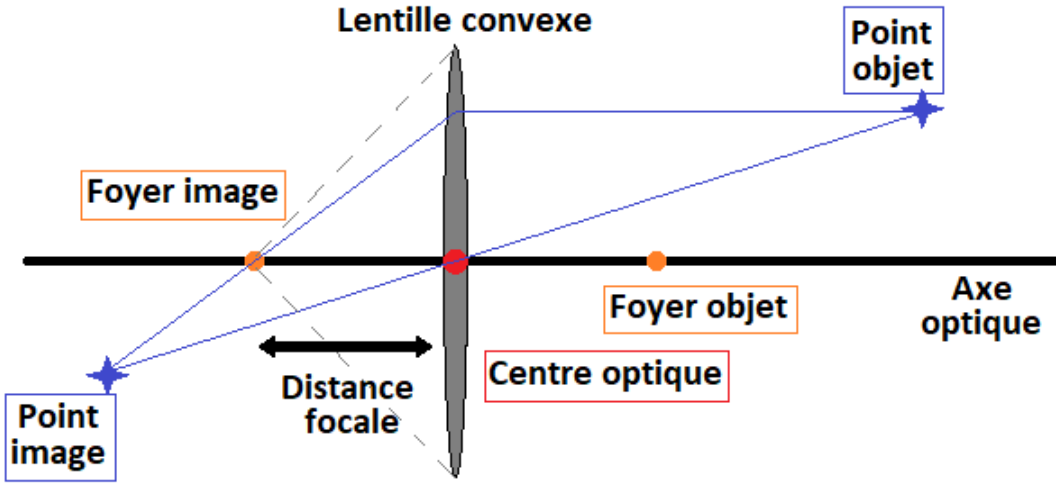

FIGURE 2.14 – Schéma de la formation d'une image

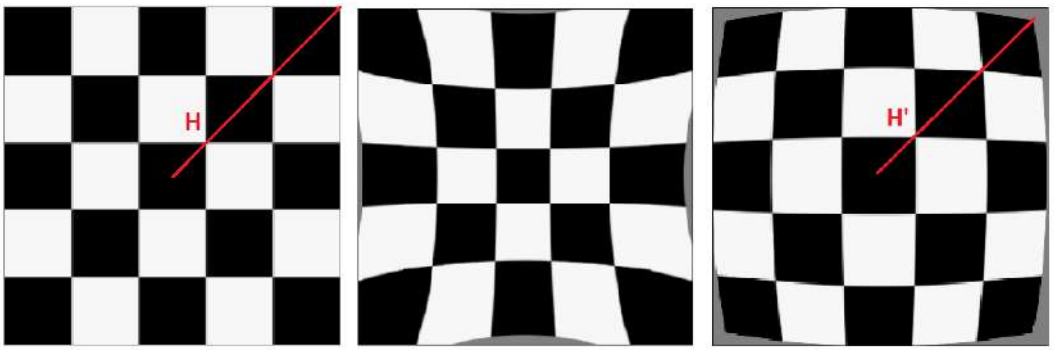

FIGURE 2.15 – Distorsions d'une image. À gauche, l'image initiale. Au centre, une distorsion de l'image en coussinet. À droite, une distorsion de l'image en barillet

La distorsion tangentielle s'écrit :

$$x_{distorded} = x\left(2p_1xy + p_2(r^2 + 2x^2)\right)$$

$$y_{distorded} = y\left(2p_2xy + p_1(r^2 + 2y^2)\right)$$

Avec : $r = \sqrt{x^2 + y^2}$

Il faut donc estimer les 5 coefficients $coeff = (k_1, k_2, k_3, p_1, p_2)$ qui permettent de retrouver (x, y) à partir de $(x_{distorded}, y_{distorded})$. Cette estimation se fait à l'aide d'une mire généralement représentée par un damier (BARRETO et al., 2009). La connaissance a priori de la forme du damier combinée à la détection des intersections des cases noires avec les cases blanches permet de mesurer les distorsions de l'image et ainsi d'estimer les paramètres précédents utiles à la correction de cette image. À noter que cette opération nécessite plusieurs

captations de la mire dans des positions différentes afin d'estimer au mieux les paramètres (Figure 2.16). Une fois cette distorsion corrigée, les éléments courbés redeviennent droits. Cela permet d'avoir des mesures homogènes dans l'ensemble de l'image. Par exemple, pour une caméra placée à l'aplomb ayant un angle de visée de 90° avec le sol, 50cm au centre de l'image est égal à 50 cm sur le bord de l'image. Cette affirmation n'est pas vraie avant la correction de la distorsion. Une analyse des surfaces de 324 poulets de 26 jours d'âge vues de dessus avec une caméra à 4 mètres de hauteur en fonction de leur distance au centre de l'image donne :

$$S = 2,69.10^{-2} * D + 8969$$

Avec S la surface d'un animal et D sa distance au centre de l'image.

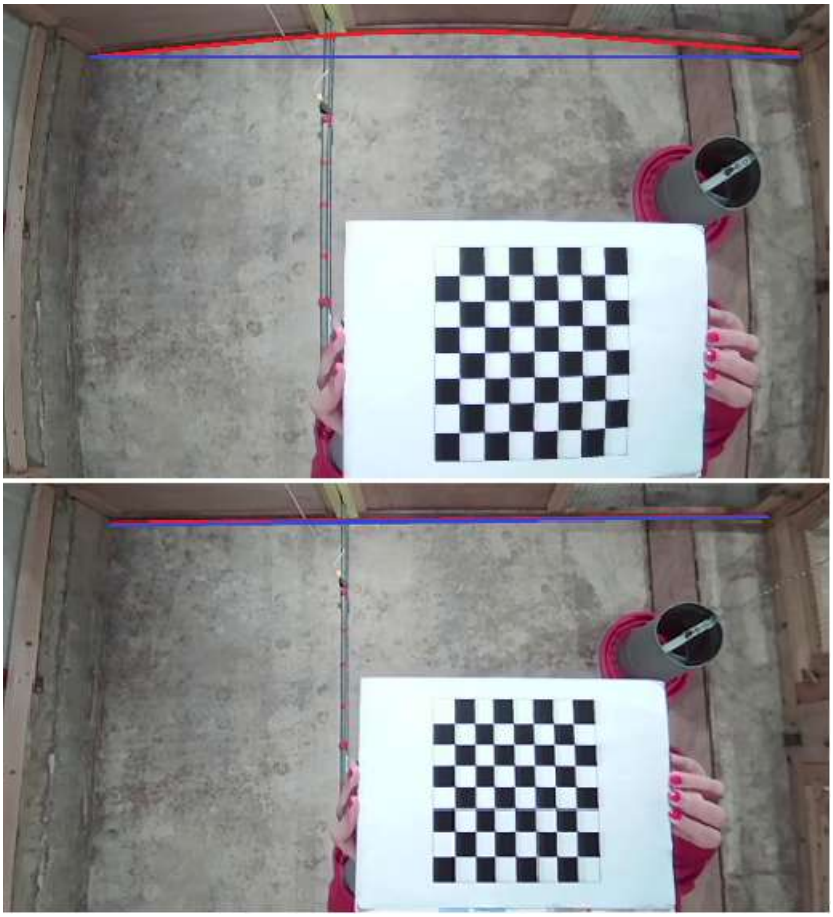

FIGURE 2.16 – Correction de la distorsion par une mire. La courbe rouge au sol (image du haut) est bien transformée en droite dans l'image du bas.

2.2.3 Réseau de caméras

Analyser le comportement des animaux implique entre autre qu'il faut être capable de connaître les distances parcourues par les animaux. Bien qu'une

caméra filmant le sol de façon inclinée (voire Figure 2.17) puisse filmer un champ au sol plus large qu'une caméra placée à l'aplomb (voire Figue 2.18), celle-ci ne conserve pas les distances de façon homogène dans toute l'image à cause de la perspective induite par l'inclinaison de la caméra. C'est pourquoi toutes les caméras sont placées à l'aplomb afin de filmer le sol sans créer de perspective due à la profondeur de champ. Dans un telle situation, le champ de vision de la caméra est alors réduit et il peut être nécessaire de créer un réseau de caméra afin de couvrir une surface au sol bien plus grande.

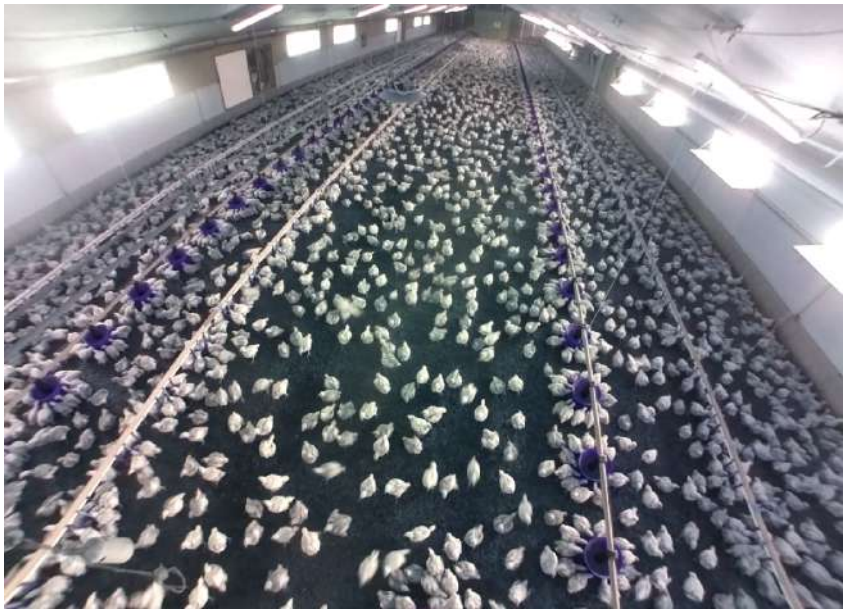

FIGURE 2.17 – Exemple de captation d'une image non placée à l'aplomb. La perspective implique un resserrement des distances en fonction de la profondeur

A ce moment-là, les caméras font place à certaines contraintes. Il y a une contrainte de hauteur et de définition de caméra et une contrainte sur l'espacement entre les caméras.

2.2.3.1 Hauteurs et définitions

Il est nécessaire de rappeler certains termes techniques liés à l'imagerie. La *définition* d'une image correspond au nombre de pixels de la caméra. Elle est exprimée en pixel. La *résolution* d'une image correspond à précision d'un pixel. Elle est généralement exprimée en pixels par pouce. Plus le nombre de pixels par pouce est grand plus la résolution est élevée. La résolution dépend à la fois de la définition de l'image et de la distance de l'objet filmé à la caméra. Créer une seule et grande image à partir d'un réseau de caméra nécessite de recouper des images ayant la même résolution, afin de garantir la conservation des distances en passant d'une image à l'autre. La figure 2.19 représente le schéma d'un réseau de caméras avec différentes résolutions. Bien qu'en faisant une estimation de la variation de résolution d'une caméra à l'autre il soit possible de créer un système de mise à l'échelle des distances en fonction

FIGURE 2.18 – Exemple de captation d'une image placée à l'aplomb. Il n'y a pas de perspectives, les distances sont homogènes sur l'ensemble de l'image

de l'endroit de l'image, cela reste très compliqué notamment s'il faut tenir compte des contraintes de distances entre les caméras présentées dans la section suivante. Pour résumer, soit les caméras sont placées dans le sens de la longueur du bâtiment, auquel cas il est plutôt facile de les fixer au plafond et elles sont toutes à la même hauteur, soit elles sont placées dans le sens de la largeur, ce qui implique qu'il faut mettre en place un dispositif pour avoir toutes les caméras à la même hauteur. Pour information, le faîte du toit au centre du bâtiment est à peu près à 5 mètres de haut, alors que le toit au niveau le plus bas est de l'ordre de 2,5 mètres. Placer les caméras dans le sens de la largeur implique de les avoir à faible hauteur. Une caméra placée à 5 mètres de haut (hauteur maximale) avec une largeur de champ de 92° ne peut filmer toute la largeur du bâtiment. Dans la mesure où le matériel (mangeoire et abreuvoir) est aligné dans le sens de la longueur, toute translation des prises de vue dans le sens de la longueur n'apporte que très peu de nouvelles configurations du comportement des poulets (en partant du principe que celui-ci est lié à leur environnement). C'est pourquoi l'installation des caméras dans le sens de la largeur des bâtiments pour les tests a été préférée, quoique plus compliquée à mettre en place d'un point de vue matériel (même hauteur pour toutes les caméras).

2.2.3.2 Espacement entre caméras

La seconde contrainte est liée à l'espacement entre les caméras. Le recoupement des images entre les caméras permet de créer une sorte d'image panoramique. Avoir un champ de caméra au sol parfaitement juxtaposé ne suffit

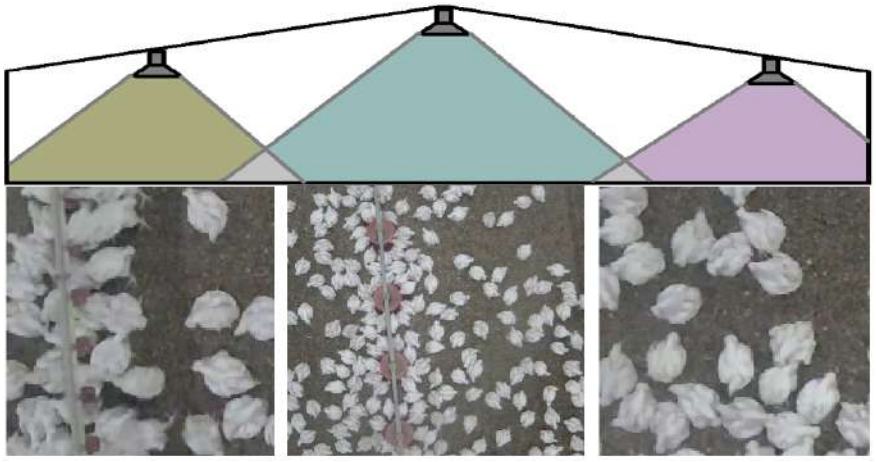

FIGURE 2.19 – Schéma d'un réseau de caméras avec différentes résolutions. Il n'y a pas conservation des distances

pas pour détecter tous les animaux (Figure 2.20, image du haut). En effet, bien que le sol puisse paraître parfaitement continue sur l'image panoramique, un animal passant du champ d'une caméra à l'autre disparaîtrait entre les deux. L'idéal est d'avoir la hauteur maximale de recouvrement (zone foncée sur l'image du bas 2.20) égale à la hauteur maximale d'un poulet enfin de ne pas, voire disparaître un animal qui se déplace du champ d'une caméra à une autre.

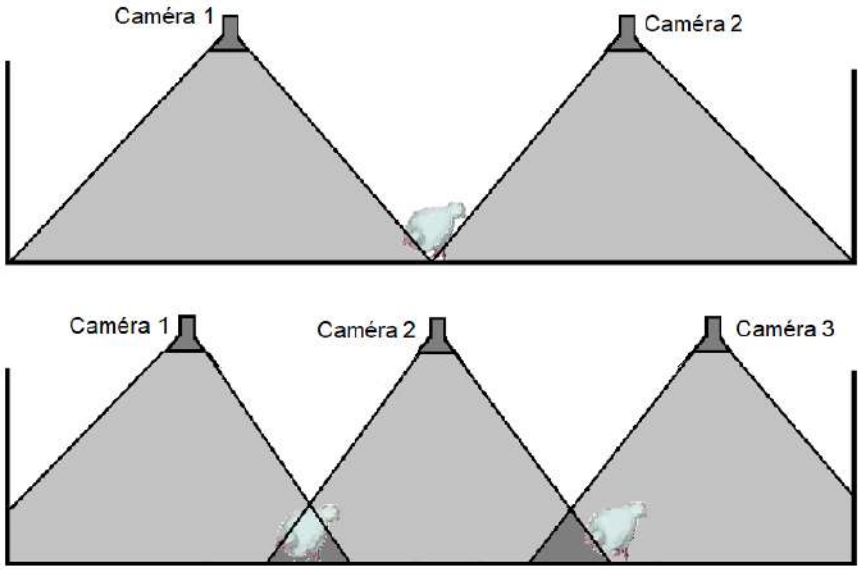

FIGURE 2.20 – Image du haut : Caméras trop éloignées, le poulet disparaît. Image du bas : Espacement optimal, la hauteur de la zone recoupée fait la taille d'un poulet

Un vrai panoramique consiste en une rotation de la caméra, ce qui ne laisse pas un objet être filmé de plusieurs angles différents. Ici, la transformation qui permet de passer de l'image d'une caméra à une autre correspond à une translation. La différence d'angle de vue d'un poulet vu par deux caméras est

d'à peu près 90°. Cela veut dire que dans une telle situation l'animal n'a pas du tout la même représentation sur l'image de chaque caméra.
L'assemblage d'images panoramiques peut se découper en quatre étapes :
— Détecter des descripteurs (section 2.1.1.3) dans chacune des images. Les descripteurs les plus utilisés pour ce type d'opération sont les descripteurs SIFT 2.3).
— Rechercher les descripteurs communs à plusieurs images.
— Définir la transformation à appliquer à une image afin d'avoir ses descripteurs qui puissent matcher avec ceux de l'autre image.
— Juxtaposer l'image transformée avec l'image suivante.
Malheureusement cette façon de procéder est difficilement applicable dans notre situation.
— Si l'assemblage d'images se base sur des descripteurs représentant majoritairement le sol, les images risquent de ne pas être suffisamment recoupées et des poulets sont alors perdus entre les champs de deux caméras.
— Les points communs aux images à assembler sont vus par des angles différents de près de 90°, ce qui entraîne la création de descripteurs bien différents.
— Cette différence d'angle empêche également de retrouver une image panoramique bien unie.
Afin de perdre un minimum d'information, la zone de recouvrement est alors définie comme une moyenne des zones de recouvrement des deux caméras (Figure 2.21). Les paramètres de cet ajustement sont fixés expérimentalement. Une fois fixés pour une mise en place de caméras donnée, ceux-ci restent alors inchangés.

2.2.4 Synchronisation

Quoique les différentes caméras soient pilotées par le même enregistreur, il se trouve que le début d'enregistrement des caméras n'est pas synchronisé. Cela implique qu'un animal sortant du champ d'une caméra peut mettre jusqu'à une seconde avant d'apparaître dans celui de la suivante, ce qui est pénalisant lorsque l'on cherche à faire du suivi de poulets. Une méthode automatique de synchronisation a été mise en place afin de résoudre le problème. Une corrélation est faite entre les zones de recouvrement de deux caméras. Le but est d'atteindre une valeur de corrélation maximale entre ces deux zones en variant le temps de début d'enregistrement d'une des deux images. Bien que ce système de synchronisation fonctionne bien la majorité du temps, le taux de réussite n'est pas parfait. En effet, la corrélation est faite entre deux images prises d'un angle de vue différent et l'auto focus des caméras perturbe également cette recherche de maximum de corrélation. Lorsque la synchronisation n'est pas réussie, celle-ci doit être ajustée à la main.

FIGURE 2.21 – Exemple d'image panoramique à partir de 3 caméras. Les deux bandes floues correspondent aux deux zones de recouvrement

2.2.5 Détection du matériel

La détection de la position des mangeoires et des abreuvoirs est une étape de traitement qui n'a besoin d'être effectuée qu'une seule fois en début de vidéo. Les mangeoires et les abreuvoirs sont des éléments essentiels de l'environnement des poulets. Leurs positions sont utilisées pour l'analyse du comportement des animaux. Puisque la position des abreuvoirs vient à varier en cours de lot (la hauteur des abreuvoirs est adaptée à la hauteur des poulets), celle-ci doit être recherchée à chaque début de traitement d'une vidéo.

La première étape consiste à transformer l'espace des couleurs RGB en HSV (image A sur la Figure 2.22). Les mangeoires étant de couleur rouge ou jaune (seules couleurs rencontrées sur les analyses vidéos effectuées sur 4 élevages différents), le but est alors de rechercher les zones de l'image à forte saturation de couleur (image B sur la Figure 2.22). Puis des opérations d'érosion et de dilatation (ref section **??**) sont appliquées sur l'image a forte saturation afin de retourner les différents objets de l'image (image C sur la Figure 2.22). Les plus gros objets correspondent alors aux mangeoires et les plus petits aux abreuvoirs. Chaque type d'objet est classé selon sa taille (les plus gros pour les mangeoires et les plus petits pour les abreuvoirs). Chaque objet est représenté par un ensemble de pixels appelé blob. Le centre de chaque mangeoire est représenté par le barycentre de chaque blob. Le diamètre des mangeoires est une valeur commune à toutes les mangeoires égale au diamètre maximal parmi les blobs mangeoires. Les abreuvoirs sont représentés par des droites.

Les blobs des abreuvoirs correspondent aux pipettes. Une régression linéaire est effectuée entre les pipettes afin de déterminer la droite qui passe au plus proche de chaque pipette.

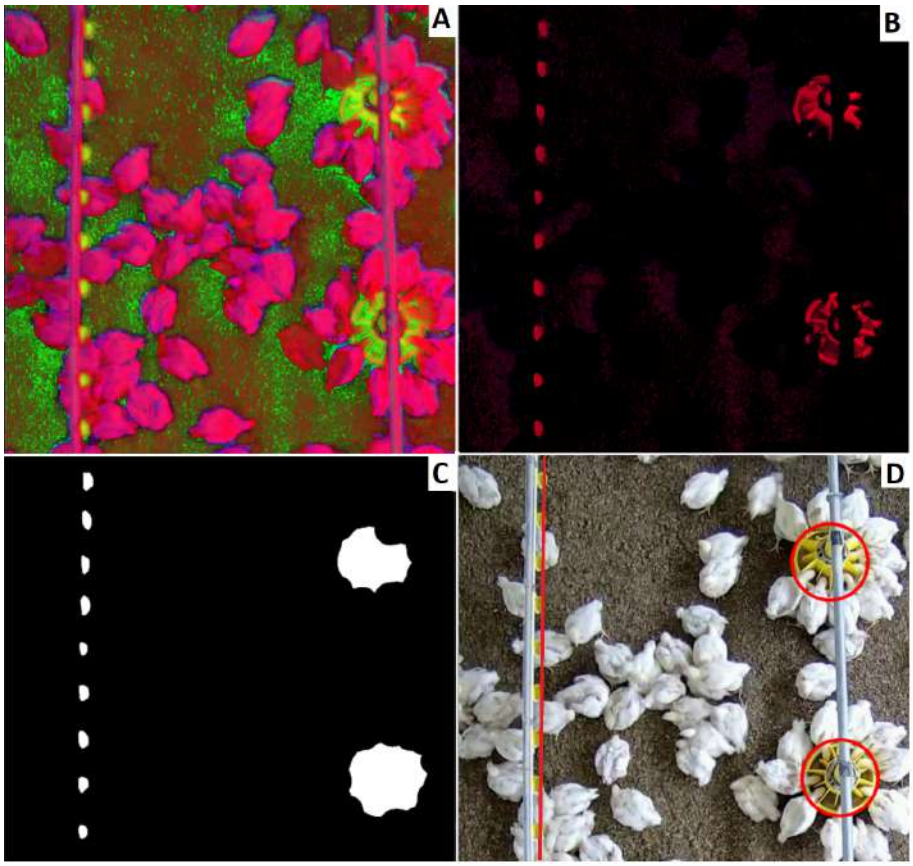

FIGURE 2.22 – Étapes de traitement de détection des mangeoires et des abreuvoirs. Image A : Transformation du domaine des couleurs. Image B : Détection des zones à forte saturation de couleurs. Image C : Dilatation et érosion de l'image. D : Détermination de la position des mangeoires et des abreuvoirs.

2.3 Méthodes testées

2.3.1 Méthodes classiques de traitement d'images

Une première méthode de détection des animaux est basée sur des outils classiques de traitement d'images. S'il est possible de détecter tous les animaux individuellement sans passer par un réseau neuronal, c'est alors un gain de temps de calcul et de moyen (les réseaux de neurones convolutifs ayant besoin de GPU afin de tourner efficacement). Les outils nécessaires à la détection sont ceux présentés dans la section 2.1.2.
Cette détection se déroule en plusieurs étapes :

— **Changement de domaine couleur :** L'espace couleur est transformé du domaine (Rouge, Vert, Bleu) en (Teinte, Saturation, Intensité).

— **Seuillage :** Sur les images tests, les poulets étant plus clairs que la litière, seuls les pixels de forte intensité sont conservés identiques tandis que les autres sont forcés à 0 (Figure 2.23, image B). Cela peut fonctionner car le poulet ROSS 308 est un poulet assez blanc qui se démarque des litières quelles qu'elles soient, d'autant plus qu'elles sont vite salies après seulement quelques jours.
— **Opérations morphologiques :** Une opération d'érosion puis dilatation (ouverture) est appliquée afin d'éliminer les objets détectés de trop petite taille (Figure 2.23, image C).
— **Mesure de distances :** À ce moment-là, chaque pixel appartient soit à un objet, soit à l'arrière plan. On mesure alors la plus petite distance entre chaque pixel et le pixel d'arrière plan le plus proche. Cela crée une sorte de carte d'intensité (Figure 2.23, image D).
— **Centres des poulets :** Chaque maximum local de la carte d'intensité est considéré comme le centre d'un poulet. Afin de limiter les mesures aberrantes, un seuil de distance minimale entre les positions de chaque poulet est fixé. Augmenter ce seuil réduit alors le nombre d'animaux détectés. Si le nombre de poulets sous le champ de la caméra est connu d'avance, ce seuil peut être modifié jusqu'à atteindre le nombre de maximums locaux souhaité (Figure 2.23, image E).
— **Kmeans :** Chaque maximum local mesuré précédemment est alors utilisé comme centre pour l'application de la méthode des Kmeans. La méthode des Kmeans permet alors de déterminer quels pixels appartiennent à quel poulet, ainsi qu'à ajuster la position du centre de chaque animal. Les tests sur les mesures de distances sont effectués premièrement avec une distance euclidienne, puis avec une distance de Mahalanobis. Puisque les poulets ont une forme plutôt ellipsoïdale, la distance de Mahalanobis permet de tenir compte de l'étalement de la distribution des pixels formant les poulets (Figure 2.24). Pour information, la distance de Mahalanobis est définie comme :

$$D_x = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)^T}$$

Avec x la variable dont on veut mesurer la distance, μ le centre du nuage de points et Σ la matrice de covariance du nuage de points de centre μ.

Cette façon de détecter les poulets nécessite le réglage de nombreux paramètres. Le seuillage des intensités de pixels est sensible à l'éclairement. Il faut impérativement le bon seuillage pour séparer les poulets du fond, mais ce seuillage doit également pouvoir être adapté à l'environnement. L'environnement peut changer au cours du temps, comme par exemple à cause de la variation de luminosité durant la journée, ou encore la dégradation de la litière au cours du lot. L'efficacité des opérations morphologiques dépend de la taille et de la forme de l'élément structurant. Si celui-ci est trop petit, il reste alors des résidus dans l'image qui sont considérés comme des poulets alors qu'il n'y en a pas (Figure 2.24 image C). Un élément structurant trop grand risque de faire disparaître des poulets, d'en scinder en deux ou, dans une moindre mesure, de perdre des informations sur leur forme et leur taille. La

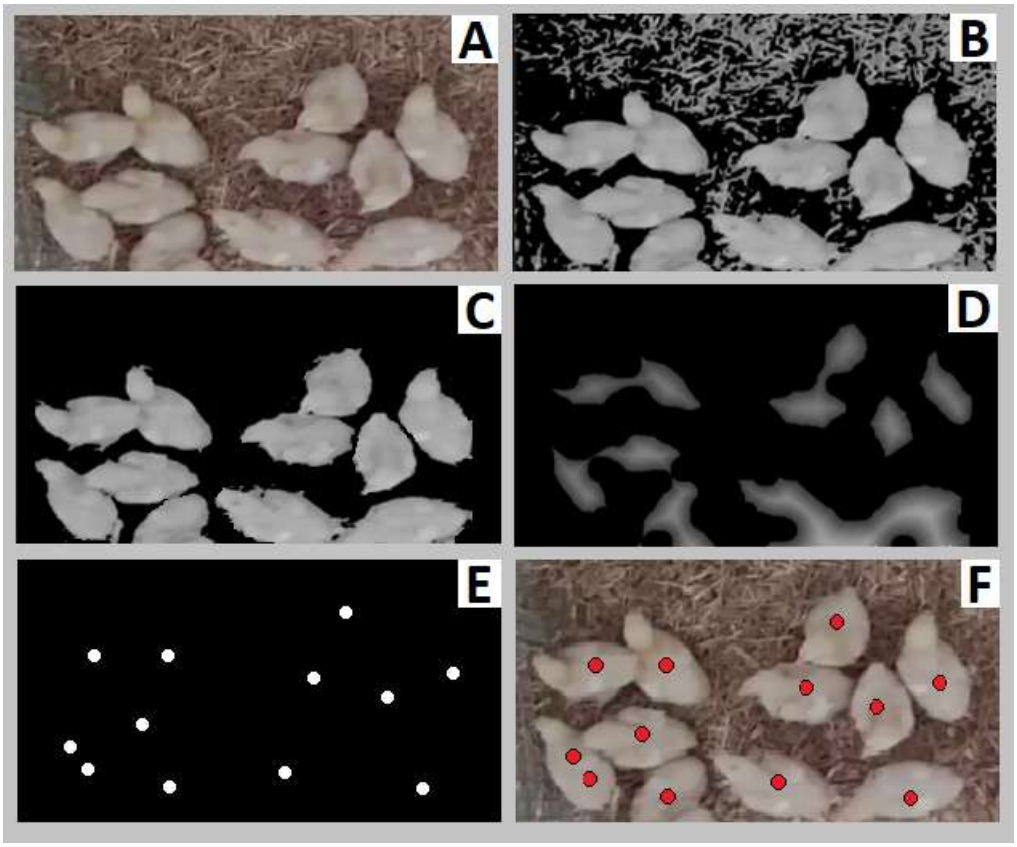

FIGURE 2.23 – A : image initiale, B : image après seuillage des intensités, C : image après l'opération morphologique, D : carte d'intensité des distances, E : image des maximums locaux, F : image finale

mesure des distances des pixels de fond ne requiert pas de paramètres particuliers à fixer. En revanche, le nombre de maximums locaux à sélectionner détermine le nombre d'animaux à détecter. La carte d'intensité de distance fait ressortir des blobs. Pour rappel, un blob est toute forme ayant une surface finie qui se détache de l'arrière plan de l'image (voir section 2.2.5). Les plus petits d'entre eux correspondent généralement à des poulets uniques, les plus gros correspondent à des regroupements de poulets. Les plus petits donnent alors une information sur la taille d'un poulet. Cela peut permettre de fixer un seuil de distance entre deux maximums locaux en utilisant par exemple un pourcentage du plus petit diamètre du blob. La Figure 2.24 illustre avec l'image B une segmentation suite à une bonne estimation du nombre de maximums locaux. L'image D de cette même figure représente une segmentation suite à une mauvaise estimation du nombre de maximums locaux. Les points rouges correspondent aux positions des maximums locaux.
Mais la principale limite de ce type de méthode ne vient pas des paramètres à régler. Même si les méthodes qui permettent de fixer ces paramètres manquent de robustesse, il est toujours possible de les fixer expérimentalement afin de d'obtenir de bons résultats. Le défi est d'être capable de discerner des poulets au milieu de gros regroupements. C'est d'ailleurs pour cela que la plupart des méthodes classiques de détections de poulets mentionnées dans la

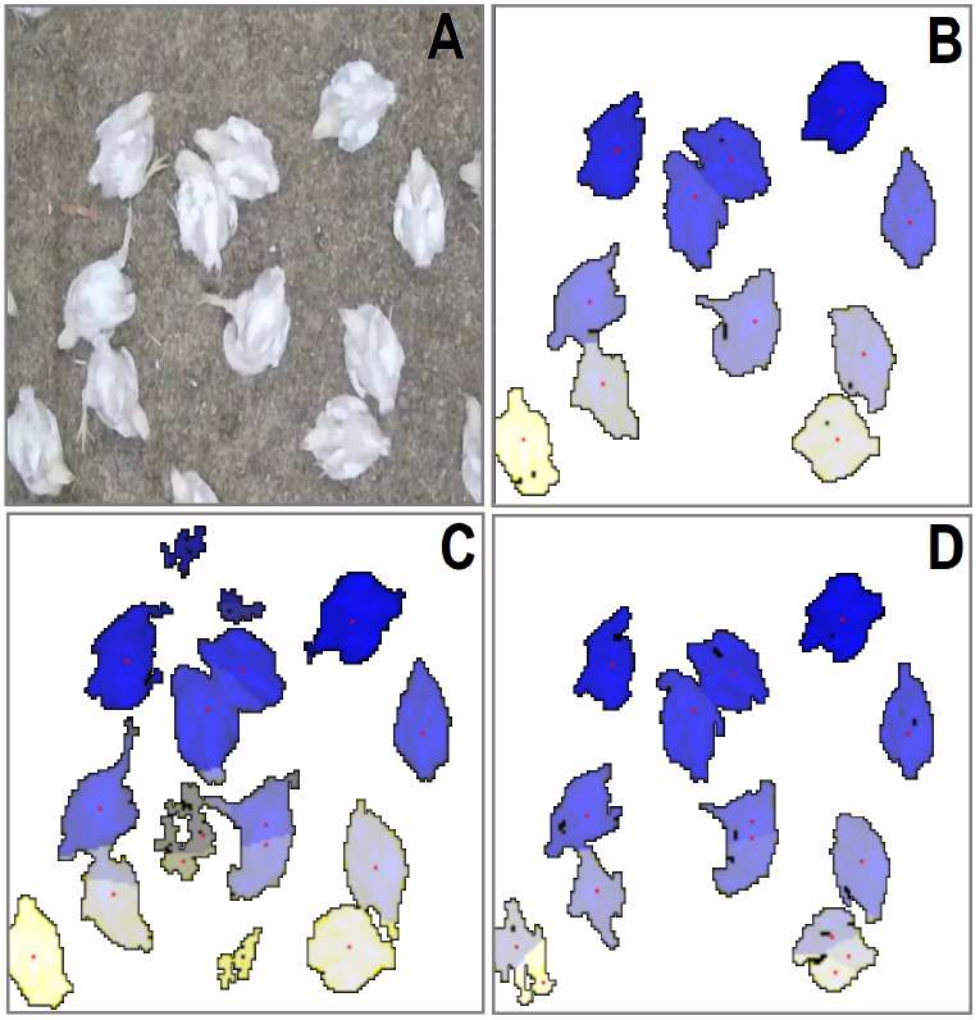

FIGURE 2.24 – Segmentation des poulets pas la méthode des K-means. A : image initiale, B : segmentation pour une valeur optimale de positions initiales, C : segmentation d'une image dont l'opération de morphologie est mal faite, D : image dont de nombre de positions initiales est trop important

section **??** ne s'appliquent qu'à des petits lots de poulets en conditions expérimentales. La méthode de détection précédemment citée est limitée à des regroupements de l'ordre de 5-6 poulets. Au-delà de 6 poulets, la carte d'intensité des distances est formée d'un très gros blob ne comprenant que très peu de maximums locaux. Dans un premier temps, il a été question d'estimer le nombre de poulets présents dans un blob en comparant la surface de ce blob et de celle de référence correspondante a des poulets isolés. Cette technique permet d'estimer assez efficacement des regroupements toujours d'un même ordre de grandeur d'à peu près 6 poulets. Au-delà de 6 poulets, plus les regroupements comportent un nombre élevé de poulets, plus ceux-ci peuvent être resserrés, ce qui implique que la surface du blob correspondant ne permet plus d'estimer le bon nombre de poulets. Enfin, la méthode

des Kmeans permet ici de regrouper des pixels autour de centres de poulets. Puisque au milieu d'un regroupement de poulets la forme de ces poulets n'est pas claire, la segmentation se traduit par des frontières toutes droites correspondant à des médianes entre les centres des poulets. Le poulet en bas à droite de l'image D de la figure 2.24 peut représenter un blob à 3 maximums locaux. Les trois formes issues de cette détection sont limitées par des droites. C'est exactement ce que l'on retrouve sur de gros regroupements de poulets. On peut tester tous les paramètres possibles, toutes les méthodes de segmentations voulues, les méthodes classiques de traitement d'images sont vite limitées dans le cas de la détection de poulets en conditions commerciales lorsque l'on fait face à des regroupements de centaines de poulets. Une solution peut alors être trouvée grâce aux méthodes d'apprentissage et notamment aux réseaux neuronaux convolutifs. La section suivante présente la méthode retenue pour la détection des poulets.

2.3.2 Détection par apprentissage profond

Bien entraîné, un réseau de neurones est capable d'effectuer des tâches très complexes. Cela va de la segmentation sémantiques où chaque pixel est affecté à une classe (Figure 2.25 image B), à la segmentation d'instance où chaque pixel est affecté à un objet unique au sein d'une classe (Figure 2.25 image C), en passant par la localisation (Figure 2.25 image D). Une fois entraîné avec une large base de donnée couvrant le maximum de situation possibles (plusieurs éclairements, plusieurs types de litières, plusieurs tailles de poulets ...), un réseau de neurones convolutifs (CNN) est alors capable de détecter plus facilement les poulets au sein d'un gros regroupement qui est une limite aux outils de traitement d'image classique (section **??**). Les conditions idéales que doivent remplir un CNN sont : détecter parfaitement tous les objets présents dans l'image, les détecter le plus rapidement possible et être facilement adaptable à toutes tailles d'image d'entrée. Enfin, dans le cadre d'un tracking de poulets, la principale attente des performances du CNN reste la localisation des animaux. Être capable en plus de la localisation d'avoir une bonne segmentation d'instance des animaux ne serait que du bonus. C'est pourquoi l'objectif premier des performances du CNN correspond à une bonne localisation des animaux.

2.3.2.1 Pré-tests et sélection du réseau

L'objectif est de mesurer des indicateurs de comportement grâce au tracking de poulets. Dans la mesure où de nombreux CNNs sont accessibles en open source, ce chapitre n'a pas pour but de développer un CNN pour la détection des poulets, mais plutôt d'en trouver un déjà existant le plus adapté à la détection de poulets.

Le modèle Faster R-CNN Un CNN effectuant de la détection d'objet recherche d'une part les zones de l'image dans lesquels se trouvent les objets

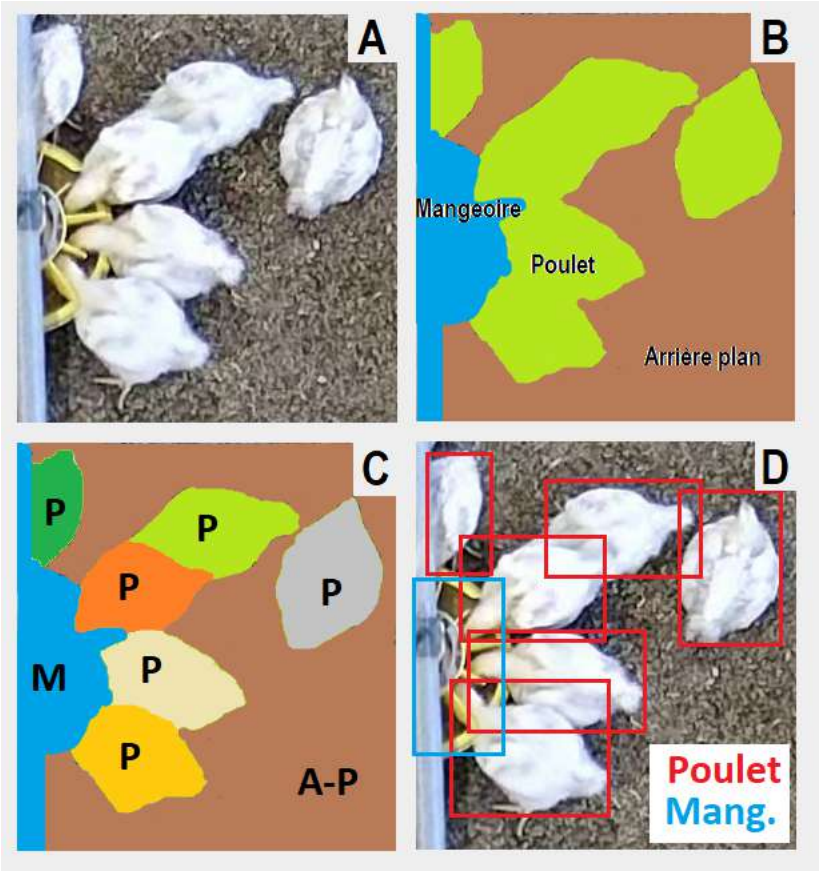

FIGURE 2.25 – Différents types d'opérations faisables par un réseau de neurones convolutif. A : image initiale, B : segmentation sémantique, C : segmentation d'instance (P=Poulet, M=Mangeoire, A-P=Arrière-plan), D : localisation

puis classifie ces objets tout en ajustant leur localisation. Si ces deux étapes sont effectuées séparément, on appelle ces réseaux de neurones des *two-stage detectors*. D'un autre côté ceux qui effectuent ces deux étapes en même temps sont appelés *one stage detectors*. La différence entre ces deux types de détecteurs se trouve dans le compromis qui est fait entre la justesse de détection (bien détecter tous les objets avec une bonne localisation) et le temps d'exécution. Globalement les *one stage detector* comme le modèle YOLO (You Only Look Once) (REDMON et al., 2016) sont plus rapide et les *two stage detectors* plus précis (CARRANZA-GARCÍA et al., 2021). Les modèles *two-stage detector* ont l'avantage d'être plus flexibles sur la taille des images d'entrée et certains d'entre eux sont présentés comme étant capable de travailler en temps réel. Parmi ces *two stage detectors* on retrouve : RCNN (Region-based Convolutional Neural Network) (GIRSHICK et al., 2014) et SPPNet (Spatial Pyramid Pooling network) (HE et al., 2015) en 2014, Fast RCNN (GIRSHICK, 2015) et Faster RCNN (REN et al., 2015) en 2015, Mask R-CNN (HE et al., 2017) et FPN (Features Pyramid Networks) (LIN et al., 2017) en 2017. Le modèle Faster RCNN est le dernier modèle de la famille des RCNN. Il est alors en 2015 présenté

comme plus rapide et plus performant d'un point de vue qualité de détection (CARRANZA-GARCÍA et al., 2021). Le modèle Mask R-CNN est basée sur le modèle Faster R-CNN et possède une extension qui permet d'effectuer de la segmentation d'instance. Il est alors plus long que le Faster R-CNN, mais surtout l'apprentissage nécessite une segmentation de contour de chaque objet qui est malheureusement beaucoup trop chronophage et pas envisageable dans ce projet. Enfin, le modèle FPN permet de détecter des objets de différentes résolutions. Bien qu'il soit plus adapté aux différentes échelles de taille des animaux, son temps de traitement est extrêmement long.

L'architecture d'un modèle Faster R-CNN est présenté Figure 2.26. Le premier bloc correspond à la détection des features (Features Extractor Network). C'est un réseau de neurones convolutif comme présenté dans la section 2.1.3.2. Ce CNN est appelé *backbone*. Parmi ces backbones on retrouve entre autres le modèle inception V2 (SZEGEDY et al., 2016) et le modèle ResNet-50 (HE et al., 2016). Les modèles *backbone* se différencient par leur architecture et le nombre de couches. Ces différences jouent sur le nombre de poids du modèle, le temps d'exécution et la précision des features détectées. Plus le réseau a de couches de convolutions, plus il y a de poids. Un modèle inception v2 fait à peu près 50Mo là où un resnet 101 en fait 350Mo. Plus de poids entraîne plus d'opérations de convolutions et donc un temps d'exécution plus long. D'un autre côté, plus de features peuvent être apprises lorsque le nombre de couches augmente. Ces réseaux sont des réseaux pré-entraînés. Ce sont des réseaux entraînés sur des millions d'images et des centaines de classes d'objets différentes. Il est ainsi plus rapide à un réseau d'apprendre les features qui caractérisent un poulet lorsque celui-ci sait déjà reconnaître des oiseaux. L'utilisation de réseau pré-entraîné s'appelle le transfert d'apprentissage. Un résumé de ces techniques est présenté par Pan dans son article(PAN et YANG, 2009). Des modèles comme inception v2 sont entraînés sur des bases de données comme ImageNet (DENG et al., 2009) ou COCO (LIN et al., 2014). Ces bases de données comprennent des centaines de milliers d'images, des millions d'objets labélisés pour des centaines de classes d'objets différentes. Ces réseaux pré-entraînés servent juste de point de départ pour l'apprentissage des features de n'importe quel objet. Une base de données spécifique des objets étudiés est alors nécessaire afin de focaliser l'apprentissage des features sur ces objets et être ainsi plus précis. Ce réseau *backbone* en entrée du modèle Faster R-CNN produit alors une carte d'intensité des features de poulets.

Cette carte d'intensité des features est ensuite utilisée afin d'alimenter le réseau RPN (Region Proposal Network). C'est un réseau convolutif dont les noyaux de convolutions sont fixés à des tailles 3×3. Il permet de déterminer les zones de la carte d'intensité qui correspondent à des poulets. Le résultat de cette convolution est ensuite mis à l'échelle de l'image d'origine. Sur cette image origine sont alors déterminées des fenêtres (appelées *ancres*). Il s'agit de zones rectangulaires dont les dimensions sont paramétrées en amont et

qui sont définies de telle sorte à correspondre à la dimension des objets à détecter. Un score de présence d'objet est affecté à chaque ancre proposée. Ce score est de un si un objet est bien présent dans l'ancre proposée (*foreground*), et est de zéro si l'ancre correspond à une partie de l'arrière-plan (*background*). Il n'est pas encore question de véritable classification à cette étape (savoir à quelle classe appartient l'objet) mais seulement de présence d'objets à détecter. S'ensuivent deux réseaux parallèles. Le premier de noyau de convolution de taille 1×1 fixe un score de présence d'objet dans l'image pour chaque *classe* (*foreground* et *background*). La sortie est de taille $2 \times k$, où k est le nombre d'ancres pour chaque position dans la carte d'intensité. Le deuxième de même taille de noyau retourne les dimensions des fenêtres (x et y) pour la position, h et w pour les dimensions. La sortie est de taille $4 \times k$. Un premier filtre permet de ne garder qu'une partie de ces propositions afin d'alléger les temps de traitement en se basant sur le score de présence d'un objet dans chaque ancre (*foreground* classe). Dans la mesure où plusieurs ancres proposées peuvent appartenir à un même objet, la méthode NMS pour *Non-Maximum Suppression* permet alors de filtrer les ancres proposées à retenir. Pour cela, la méthode NMS mesure le chevauchement (IoU — Intersection over Union) qu'il existe entre les différentes ancres. Si des ancres ont un score IoU supérieur à un certain seuil fixé en amont, alors elles partagent un même objet avec d'autres ancres proposées. Parmi ces ancres au score IoU élevé, sont gardées celles au score de présence le plus élevé. Cette étape correspond à la fin de la première partie du *two stages detector*.

La combinaison de la carte d'intensité des features liée aux régions proposées, fournit un ensemble d'imagettes qui sont utilisées en entrée de la deuxième partie du *two stages detector*. Chaque imagette correspond à une région bien définie de la carte d'intensité des features. Ces imagettes sont donc introduites dans le RoI Pooling (Region of Interest). Ces imagettes ayant toutes des tailles différentes, le rôle du RoI Pooling est d'obtenir une taille standard pour toutes ces imagettes. Pour des dimensions d'entrée d'une image de h par l, celle-ci est divisée en sous blocs de taille h/H et l/L (les paramètres H et L sont fixés en amont). Pour chaque bloc le RoI Pooling retient le pixel de valeur maximale afin d'obtenir en sorti une imagette de taille H par L. L'intérêt du RoI Pooling est de créer des imagettes ayant toute la même taille, ce qui permet un traitement optimisé de ces imagettes lors des opérations suivantes (classification et box régression).

À partir de là deux opérations sont effectuées dans le même style que le RPN, c'est-à-dire classifier et fixer la position et la taille des objets. Chaque imagette en sortie du RoI Pooling est mise a plat (image 2D en vecteur 1D). Chaque vecteur passe dans des réseaux full connected. Le premier possède N+1 neurones, où N correspond au nombre de classes (ici N=1, car seul la classe *poulet* est utilisée), plus 1 de présence d'objet (*foreground*). Le deuxième de 4N+1 neurones permet de d'adapter la bounding box autour de chaque objet à classer. Une fois de plus un tri est fait parmi toutes les propositions,

c'est-à-dire que sont gardées seulement celles dont le score de classe est assez élevé (classe de l'objet et classe de présence).

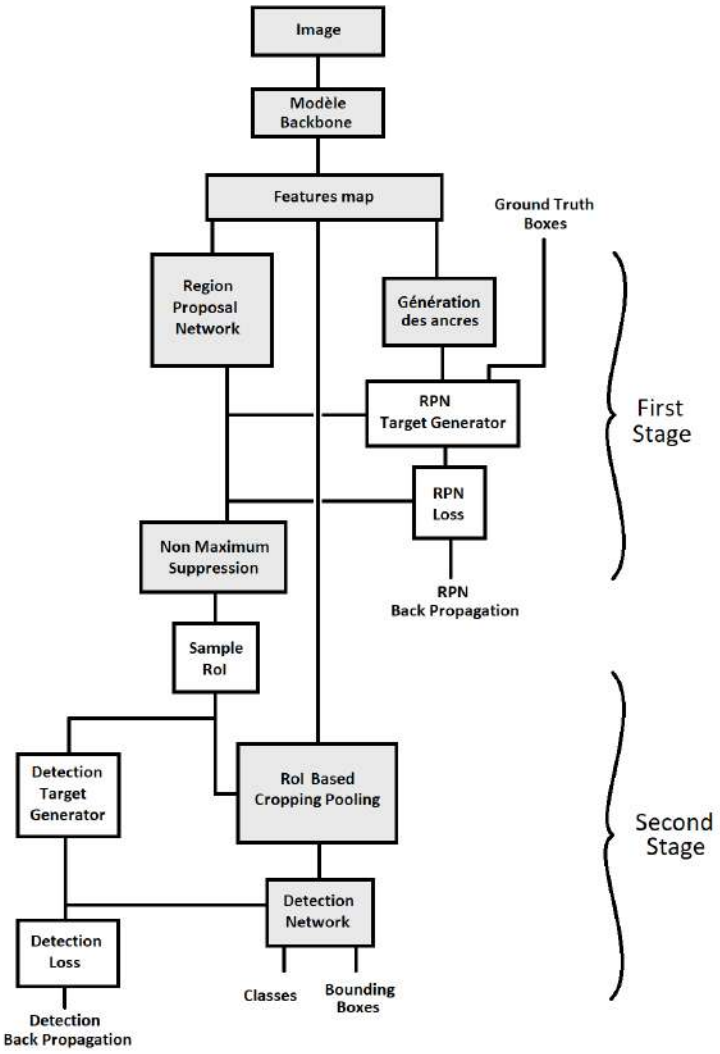

FIGURE 2.26 – Architecture d'un modèle de détection Faster R-CNN. Les blocs en blanc sont utilisés uniquement durant la phase d'apprentissage

Modèle Backbone Pour rappel, le modèle backbone est le premier CNN du modèle Faster R-CNN qui a pour but de détecter les features de l'image. Les tests ont été effectués sur les modèles suivants. Ce sont des réseaux pré entraînés accessible dans le package *Object detection* de TensorFlow.

— resnetV101 (HE et al., 2016)
— inception v2 (SZEGEDY et al., 2016)
— inception resnet v2 (SZEGEDY et al., 2017)

Le modèle inception inception V2 est présenté Figure 2.28. Il est une succession de couches de convolution 1×1, 3×3, 5×5, 7×7 et de couches de Pooling. L'intérêt du modèle inception V2, c'est qu'il optimise les opérations

de convolution en remplaçant notamment les convolutions 5×5 par deux convolutions 3×3 et les convolutions $n \times n$ par $n \times 1$ et $1 \times n$, comme présenté Figure 2.29. Cela permet de limiter le nombre de paramètres et le temps d'exécution, d'un autre côté cela limite un peu le nombre de features que le modèle peut apprendre. Un modèle ResNet est un modèle plus profond qu'un modèle inception. Le nombre de couches est généralement compris dans son nom. Par exemple un ResNet-101 possède 101 couches. Lorsque le nombre de couches est trop important, apparaît ce que l'on appelle un *problème de dégradation* (HE et al., 2016). Plus le nombre de couches augmente, plus les performences diminuent. Afin de contourner le problème, les modèles ResNets contournent certaines couches avec des blocs comme celui de la Figure 2.27

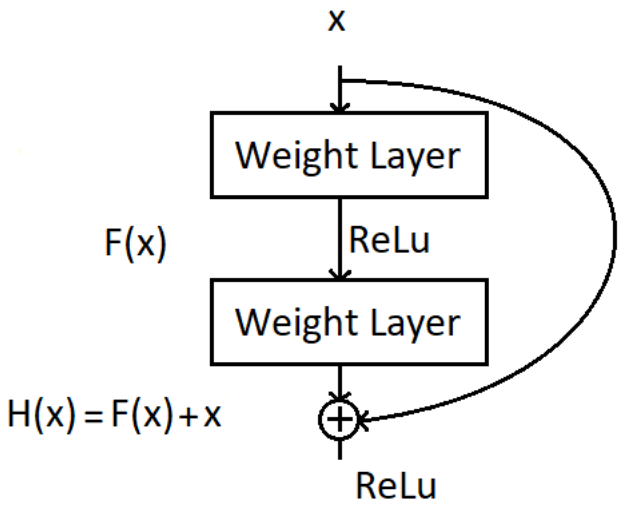

FIGURE 2.27 – Élément d'architecture d'un modèle Resnet

Les différentes caractéristiques de ces modèles backbones sont présentés dans le tableau 2.1. Les tests ont été effectués par Fromm dans (FROMM et al., 2019) sur la base de donée MS COCO. À noter que les données du tableau 2.1 mesurées à partir de la base de données MS COCO, n'ont aucune raison d'être strictement les mêmes avec notre base de données, mais cela permet de donner une idée des performances des différents réseaux. La vitesse correspond au temps de traitement d'une image entre le moment où elle entre dans le réseau de neurones et le moment où la carte d'intensité est générée. La valeur absolue de cette vitesse dépend des GPUs sur lesquels tourne le réseau de neurones. Tous les GPUS n'opèrent pas à la même vitesse. Le nombre de paramètres donne une idée de la profondeur du réseau. Plus celui-ci est profond plus le nombre de features appris augmente et permet ainsi d'améliorer la détection. En revanche, plus le nombre de paramètres est important, plus la base de données nécessaire à l'apprentissage doit être également augmentée et plus le temps de calcul augmente. La valeur de Top 1-accuracy réfère aux performances de classification. Le score MAP pour *Mean Average Precision* traduit la qualité de détection.

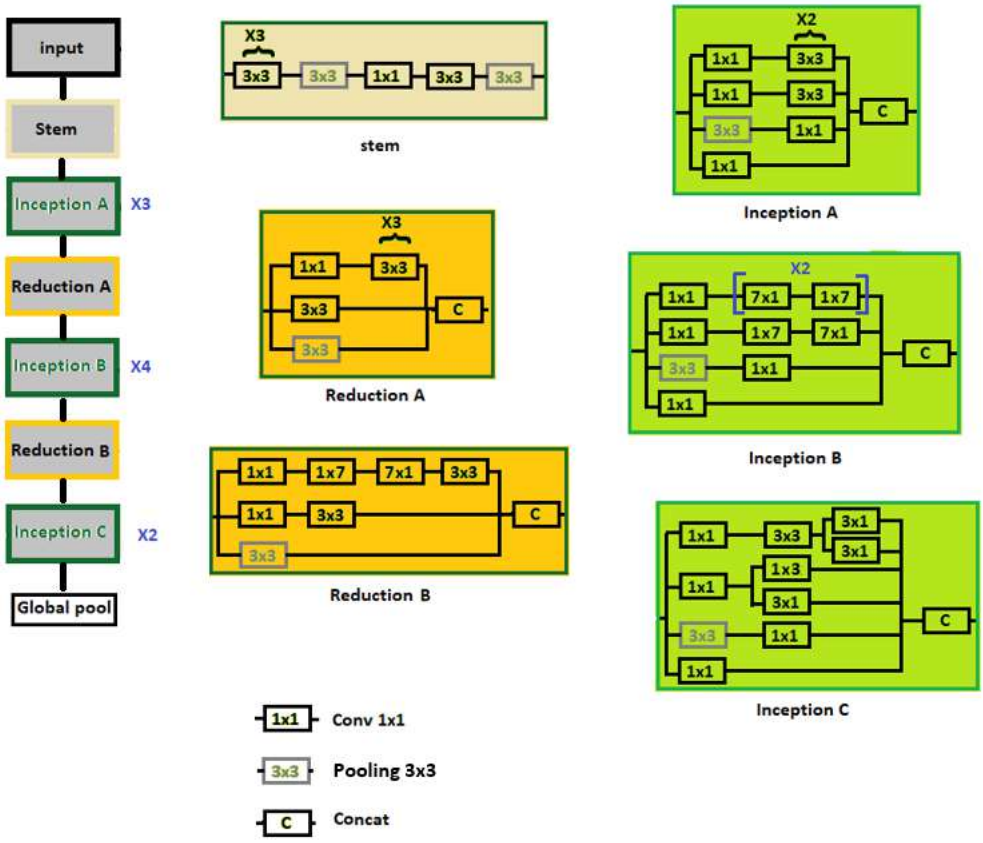

FIGURE 2.28 – Architecture d'un modèle Inception v2

Suite à des premiers tests effectués sur une base de données de ROSS 308 présentée dans la section 2.3.2.2 le modèle retenu est le modèle inception v2. Quel que soit le modèle, l'entraînement ne semble plus s'améliorer aux alentours de deux millions d'itérations. Avec un nombre de paramètres plus restreint, le modèle inception v2 atteint ces deux millions d'itérations bien plus rapidement. Le fait que la base de données soit composée d'à peu près 1000 images (seulement), explique le fait que le modèle inception v2 soit capable de détecter plus d'animaux que les deux autres modèles qui ont besoins de bien plus d'images pour fixer leurs quelque 50 millions de paramètres. Le temps d'exécution étant un paramètre non négligeable et la base de données étant relativement limitée, le perfectionnement de l'entraînement du modèle backbone s'est donc limité à celui du modèle inception V2.

2.3.2.2 Base de données

Une nouvelle base de données a été créée afin de focaliser l'apprentissage des features sur celles de poulets de chair ROSS 308. Pour chaque image, l'utilisateur encadre chaque objet présent dans le champ de l'image et assigne un label à chaque objet. Dans notre cas, on cherche seulement à labéliser une seule classe qui correspond aux poulets et qui est appelée *chicken*. In fine,

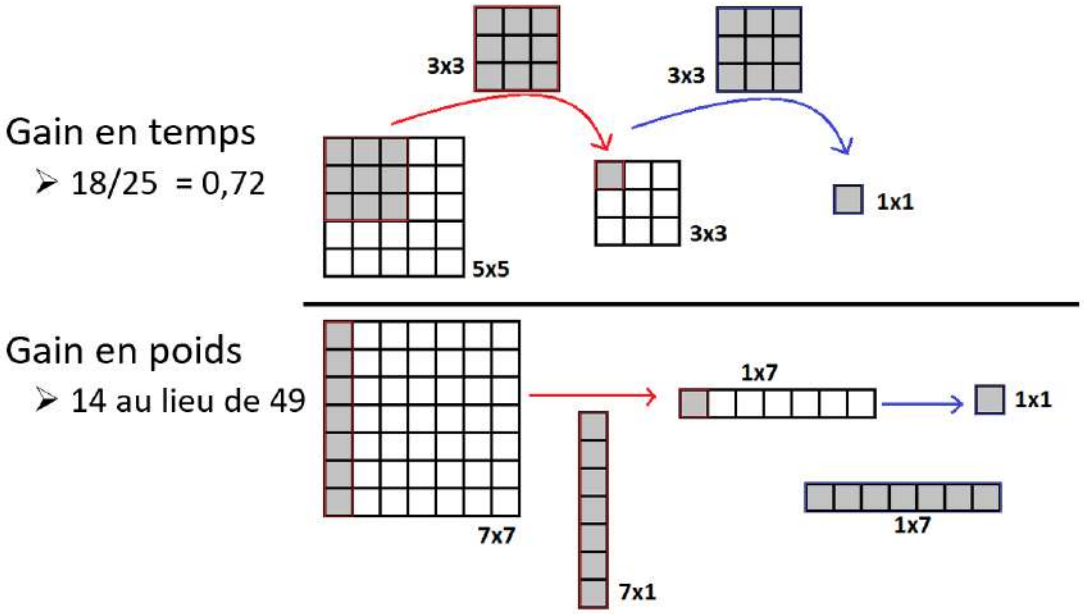

FIGURE 2.29 – Particularités du modèle Inception v2. Par rapport au modèle inception v1, les convolutions 5x5 sont transformées en deux convolutions 3x3 (gain de 1/0.72 du nombre d'opérations). De même les convolutions 7x7 sont transformées en deux convolutions (1x7 et 7x1)

TABLE 2.1 – Caractéristiques des modèles backbone testés

	vitesse (ms)	nombre de paramètres	Top 1-accuracy	MS COCO MAP
resnet 101	106	42.6M	76.4	32
inception v2	58	10.2M	73.9	28
inception resnet v2	620	54.3M	80.4	37

à chaque image est associé un fichier au format xml, dans lequel est enregistré la position de toutes les *Bounding Boxes* créée dans l'image. La Figure 2.30 montre le fonctionnement de labélisation des images. Cette base de données est composée d'une première banque d'image prise à l'INRAe de Nouzilly avec des caméras d'1M pixels. Les caméras étaient placées à 2 mètres de hauteurs au dessus de deux parquets, l'un de faible densité de poulets (12 poulets $/m^2$) et l'autre à haute densité (18 poulets $/m^2$). Chaque parquet de $12m^2$ étaient composées d'un réseau de 3 caméras. Bien qu'un modèle Faster R-CNN soit capable de prendre n'importe quelle taille d'image d'entrée, la limite de la mémoire RAM limite l'apprentissage à des images de taille 1000×1000 en entrée du réseau. Afin d'accroître la robustesse du modèle, une deuxième banque de données à été créée à partir de captations réalisées en élevage commercial, avec cette fois-ci un réseau de caméras de 2M pixels placées 2,8 mètres de haut. Cette deuxième banque d'images a été réalisée sur une litière différente de la première afin de faire varier l'arrière plan des

images. Utiliser des images prises en élevage commercial permet également d'avoir une luminosité qui varie au cours de la journée et ainsi rendre le modèle plus robuste au changement d'intensité lumineuse. Globalement, plus la base de données qui permet d'entraîner le réseau de neurones est variée, plus le réseau devient robuste.

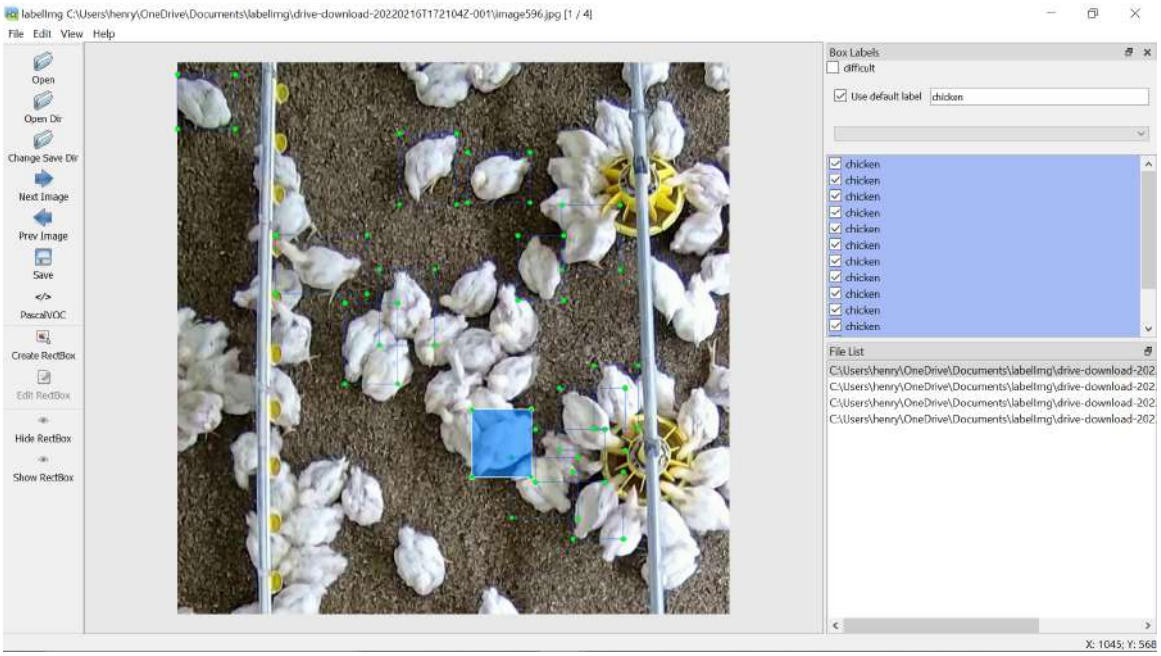

FIGURE 2.30 – Labellisation des images

Comme dit précédemment un modèle Faster R-CNN est composé d'un modèle backbone qui permet de détecter les features de l'image et de deux réseaux supplémentaires qui permettent de localiser et classifier les objets. Lors de l'entraînement du modèle Faster R-CNN, il est possible de geler le modèle backbone. C'est-à-dire qu'une fois que le modèle semble bien reconnaître les objets dans l'image, on focalise alors l'entraînement sur la classification et la localisation. Pour cela, une nouvelle base de données à été créée à partir de la première. Il s'agit de créer de nouvelles images labélisées à partir des images de la base de données d'origine. Avec une telle technique, il est alors possible de d'augmenter considérablement la taille de la base de données.

2.3.2.3 Entraînement

Les deux points essentiels à l'entraînement d'un réseau de neurones sont la base de données et le paramétrage de ce réseau de neurones. Un fichier de configuration est associé au modèle Faster R-CNN dans lequel sont répertoriés tous les paramètres sur lesquels jouer afin d'améliorer les performances du modèle en fonction du type d'objet que l'on cherche à détecter (Voir Appendix A). La suite présente les principaux paramètres et leurs impacts dans le processus de détection d'objets.

— **num classes** Permet de fixer le nombre de classes. Dans notre cas, il n'y a qu'une seule classe. Ce paramètre est donc fixé à 1.

— **image resizer** Permet de fixer la taille de l'image d'entrée. Si le modèle a été entraîné avec des images d'une certaine plage de hauteur et largeur, ce paramètre fait en sorte de retailler la nouvelle image afin que celle-ci corresponde à la taille des images qui ont entraîné le réseau. Dans la mesure où un réseau convolutif fonctionne par réduction successive des images d'entrée, si celle-ci est trop petite de base, le réseau peut être dans l'incapacité de créer une carte d'intensité. Soit le modèle fixe la taille de l'image d'entrée à une hauteur et une largeur spécifique, soit il retaille l'image en gardant les proportions de celle-ci. Les paramètres sont : min_dimension et max_dimension. Dans le cas de la fonction *keep_aspect_ratio_resizer*, un coefficient est mesuré à partir du rapport min_dimension sur max_dimension. Si le rapport petit côté sur grand côté de l'image d'entrée est plus grand que le coefficient précédemment mentionné, alors le petit côté de l'image est forcé à min_dimension et le grand côté et dimensionné afin de garder le ratio petit côté sur grand côté d'origine. Dans le cas contraire (rapport inférieur au coefficient de référence), c'est le grand côté qui est forcé à max_dimension et le petit côté ajusté en fonction. Il est possible de choisir le type d'interpolation à utiliser pour le changement de dimensions des images.
— **feature_extractor** Permet de choir de choisir le type de modèle backbone. Dans notre cas, il s'agit de faster_rcnn_inception_v2. Il est suivi de first_stage_features_stri
Ce dernier paramètre du modèle backbone correpond au ratio entre la taille d'image d'entrée et la taille de la carte d'intensité.
— **first_stage_anchor_generator** Les paramètres correspondant au _first_stage sont ceux du *Region Proposal Network*. Il s'agit ici de fixer la taille des ancres utilisées pour générer les régions d'intérêts. Ces ancres dépendent de la taille des objets dans l'image. Avec des caméras à 2 mètres de haut et des tailles de poussin à des poulets d'à peu près 40 jours d'âge, leurs dimensions varie de l'ordre d'une vingtaine de pixels à près de 200 pixels. On fixe alors la taille d'une ancre de référence, puis les différentes échelles à appliquer à cette taille de référence afin de créer d'autres tailles. Le ratio permet de jouer sur la forme des ancres. Les strides indiquent le nombre de pixels selon lesquels les ancres sont déplacées dans l'image d'origine afin de balayer toute l'image.
  — height : 128
  — width : 128
  — scales : [0.125, 0.25, 0.5, 1.0]
  — aspect_ratios : [1.0, 1.25, 0.75]
  — height_stride : 8
  — width_stride : 8

  Plus le nombre d'échelles et le nombre de ratios est important, plus grand est le nombre d'ancres créées. Un grand nombre d'ancres implique une meilleure localisation des objets, par contre cela implique un plus long temps de calcul.
— **first_stage_nms_score_threshold & first_stage_nms_iou_threshold** Ces paramètres correspondent au Non Max Suppression et au Input over Union

mentionnés dans la section 2.3.2.1. Ils permettent de filtrer une partie des propositions des régions d'intérêt. On retrouve ces même paramètres dans le second_stage. Encore une fois, moins l'on filtre de régions, meilleur est le résultat et plus long est le temps de calcul. Ces valeurs sont plus élevées dans le second_stage qui filtre alors les régions de façon plus fine.

— **first_stage_max_proposals** Ce paramètre se retrouve dans les deux stages également. La génération des ancres crée quelques dizaines de milliers de propositions (largeur carte d'intensité $\times$ hauteur carte d'intensité $\times$ nombre d'échelles des ancres $\times$ nombre de ratios des ancres). Si l'on connaît a priori le nombre maximum d'animaux dans une image, il est alors possible de régler cette valeur de proposition maximale d'objets détectés. Dans notre cas on garde 512 propositions au premier *stage* et *256* au second.

— **first_stage_localization_loss_weight & first_stage_objectness_loss_weight** Ces deux paramètres fonctionnent ensemble. On donne un coefficient de 2 au plus important et un coefficient au moins important. Dans notre cas, la localisation étant plus importante que la classification (il n'y a qu'une seule classe), le coefficient de 2 est donnée à first_stage_localization_loss_weight

— **initial_crop_size** A la sortie du *Region proposal Network* toutes les propositions sont retaillées à la même taille. Ce paramètre permet de fixer cette taille.

— **maxpool_kernel_size** Ce max pool kernel arrive juste après l'initial_crop_size. Il fixe le nombre de pixels de l'opération de pooling.

— **maxpool_stride** Ce paramètre fixe le nombre de pixels selon lesquels l'élément structurant de pooling se déplace lors de l'opération précédente. Si initial_crop_size est fixé à 12, maxpool_kernel_size et maxpool_stride à 2, alors les propositions sont réduites à des tailles de 6×6.

— **learning_rate** Le taux d'apprentissage est un paramètre de réglage dans un algorithme d'optimisation qui détermine la taille du pas à chaque itération tout en se déplaçant vers le minimum de la fonction de perte. Durant les premières itérations ce taux est fixé à une valeur relativement élevée, et décroît au fur et à mesure des itérations. On fixe ici ces taux ainsi que le nombre d'itérations auquel ils évoluent.

— **data_augmentation_options** Cette option permet des transformations aléatoires des images de la base de données qui permet d'entraîner le modèle. Par exemple dans la mesure où la caméra est placée à la verticale au plafond une transformation de rotation sur les images est tout à fait appropriée, dans la mesure où la détection ne doit pas être influencée par une rotation de la caméra. En revanche une transformation des couleurs est beaucoup moins appropriée dans la mesure où les poulets sont globalement blancs et à aucun moment l'on est amené à détecter des poulets bleus ou verts. Voici une liste des transformations que l'on peut appliquer sur les images de la base de données appliquées au poulet ROSS 308 :
  — random_rotation90
  — random_horizontal_flip
  — random_vertical_flip

- random_image_scale (selon l'âge et la hauteur de la caméra, la taille du poulet varie)
- random_adjust_brightness
- random_adjust_contrast
- random_crop_image (un animal sur le bord de l'image peut ne pas être détecté en entier)
- random_pad_image
- random_black_patches (occlusions possibles)

2.3.2.4 Utilisation du CNN

On retrouve en sortie du CNN :

- **Les bounding box** : ce sont les coordonnées des points supérieurs gauches et inférieure droit de chaque rectangle encadrant les poulets détectés. Il est alors possible d'en déduire la position centrale du poulet ainsi que ses dimensions hauteur et largeur dans l'image.
- **La classe** : le réseau associe chaque objet détecté à une classe. Dans notre cas ; la seule classe retournée par le réseau est *chicken* dans la mesure où c'est la seule étudiée.
- **Le score** : c'est une valeur entre 0 et 1 qui traduit la probabilité que l'objet détecté appartient bien à la classe associée.

Une même image passée deux fois en entrée du réseau de neurones, retourne alors deux fois le même résultat. En revanche, une légère modification de l'image d'entrée (comme une translation de quelques pixels) peut entraîner des résultats bien différents. Outre le fait que les positions des objets détectés aient alors subi une translation, certains objets non détectés lors de la première détection (et bien présent dans le champ de la caméra) peuvent alors être détecté suite à la légère modification de l'image. Inversement, des objets bien détectés la première fois, peuvent ne pas être détecté suite à la légère modification de l'image. Dans notre cas, afin de palier le problème d'objets non détectés, une transformation verticale (Figure 2.31) et une transformation horizontale peuvent être appliquées à l'image d'entrée en fonction de la qualité initiale de la détection.

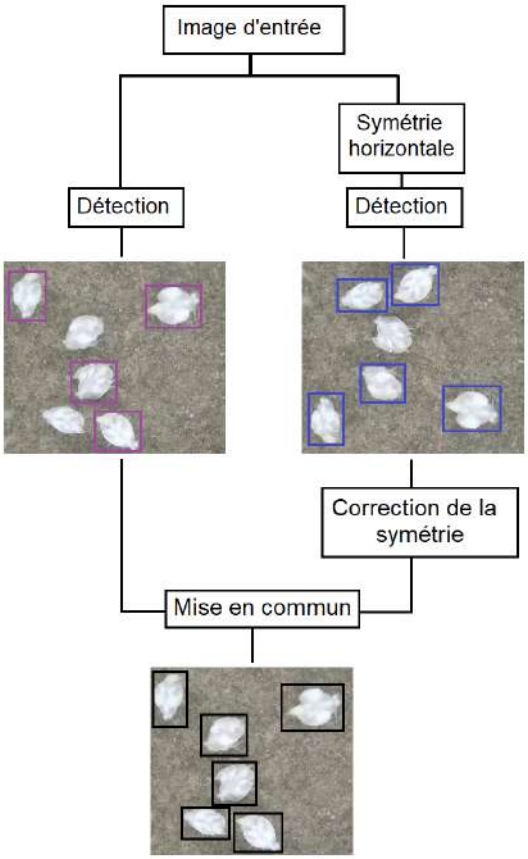

FIGURE 2.31 – Augmentation du nombre d'objets détectés grâce à une double détection de l'image d'entrée

Chapitre 3

Le suivi

3.1 Bibliographie

Il existe des algorithmes de tracking basés sur des réseaux de neurones (CIAPARRONE et al., 2020) (BI et al., 2019), ou bien des algorithmes basés sur du traitement d'images (BALAJI et KARTHIKEYAN, 2017) (KOTHIYA et MISTREE, 2015).

3.1.1 Présentation du suivi d'objets

3.1.1.1 Définitions

Le suivi d'objets est un système qui permet d'attribuer un identifiant unique à chaque objet d'une vidéo. A chaque image de la vidéo, ce système met en correspondance chaque objet présent dans la nouvelle image à ceux déjà connus dans les images précédentes. Pour simplifier, on appelle *piste* l'ensemble des détections passées appartenant à un même objet et une *cible* un objet détecté dans la nouvelle image. Le suivi met donc en correspondance une piste avec une cible et attribue à cet objet cible l'identifiant de la piste correspondante.
Une cible est caractérisée par sa position, voire par la région de l'image dans laquelle il se trouve à un instant donné. La position est constituée par les coordonnées (X, Y) de la ligne et de la colonne du centre de l'objet dans l'image. La région dans laquelle l'objet cible se trouve est généralement une zone rectangulaire ou elliptique centrée autour de la position de l'objet et circonscrite à ce même objet. On peut donc définir la position d'un objet présent sur la t^{ieme} image comme $(Z_i^t)_{i=1,..,n_t} = (X_i^t, Y_i^t)_{i=1,..,n_t}$, avec n_t le nombre d'objets sur cette t^{ieme} image. De la même façon, on peut définir la région qui comprend l'objet (*Bounding box* en anglais) comme $(R_i^t)_{i=1,..,n_t} = (H_i^t, L_i^t)_{i=1,..,n_t}$, avec H et L le grand côté et le petit côté de cette région rectangulaire.

3.1.1.2 Permutations mathématiques

Définition : Supposons dans un premier temps que le système soit fermé, c'est-à-dire que tous les objets restent dans le champ de la caméra et qu'aucun objet n'en sorte. De plus, nous supposons la détection parfaite. En d'autres termes, tous les objets présents sont détectés. Ces hypothèses garantissent que $n_1 = \ldots = n_T = N$.

Dans notre situation simplifiée (système fermé), effectuer un tracking revient à construire une suite de T permutations $(\sigma^t)_{t\leq T}$. La permutation σ^t est la permutation à appliquer aux objets de l'image $t-1$ pour obtenir les numéros des objets cibles sur l'image t.

Exemple : Nous définissons ici deux permutations σ^1 et σ^2 qui correspondent respectivement aux passages des indices du temps $t=0$ à $t=1$ et du temps $t=1$ à $t=2$.

$$\sigma^1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 2 & 4 & 1 \end{pmatrix}$$

$$\sigma^2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 3 & 1 \end{pmatrix}$$

Un objet d'indice $j=3$ au temps $t=0$ noté en Z_3^0, correspond alors à l'objet d'indice $j=2$ au temps $t=1$ noté Z_2^1 car $\sigma^1(3)=2$. De la même façon, l'objet Z_2^1 correspond à l'objet Z_5^2 car $\sigma^2(2)=5$.
On note $\sigma^2 \circ \sigma^1$ la permutation qui permet alors de passer des indices à $t=0$ aux indices à $t=2$. La Figure (3.1) représente cette évolution des indices résumée dans la matrice suivante :

$$\sigma^2 \circ \sigma^1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 2 & 4 & 1 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix} \begin{matrix} \sigma_1 \\ \sigma_2 \end{matrix} \tag{3.1}$$

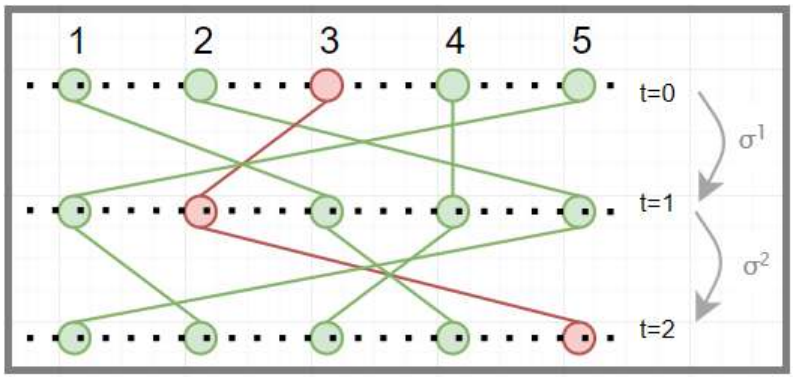

FIGURE 3.1 – Permutation du temps t=0 au temps t=2

Si l'on prend pour exemple l'évolution de l'objet d'indice 3 à l'instant 0, cela donne :

$$Z_3^0 \rightarrow Z^1_{\sigma^1(3)} \rightarrow Z^2_{\sigma^2\circ\sigma^1(3)}$$

$$Z_3^0 \rightarrow Z_2^1 \rightarrow Z_{\sigma^2(2)}^2$$

$$Z_3^0 \rightarrow Z_2^1 \rightarrow Z_5^2$$

Dans la suite, nous notons σ^{-j} la permutation réciproque de σ^j ie $\sigma^{-j} \circ \sigma^j = \sigma^j \circ \sigma^{-j} = Id$. On peut donc reprendre l'exemple précédent et remonter du temps $t = 2$ au temps $t = 0$. Sachant que $\sigma^{-2}(5) = 2$ et $\sigma^{-1}(2) = 3$, on peut alors noter :

$$Z^0_{\sigma^{-1}\circ\sigma^{-2}(5)} = Z^0_{\sigma^{-1}}(2) = Z_3^0$$

De même, nous notons σ^{-t} la permutation réciproque de σ^t de telle sorte que $\sigma^{-t} \circ \sigma^t = \sigma^t \circ \sigma^{-t} = Id$, et $\sigma^{t,1}(i)$ la forme contractée de $\sigma^t \circ \ldots \circ \sigma^1(i)$:

$$Z_i^0 \rightarrow Z^1_{\sigma^1(i)} \rightarrow \ldots \rightarrow Z^t_{\sigma^{t,1}(i)}$$

$$Z^0_{\sigma^{-1,-t}(j)} \leftarrow \ldots \leftarrow Z^{t-1}_{\sigma^{-t}(j)} \leftarrow Z_j^t$$

Dans cette dernière équation, $i = \sigma^{-1,-t}(j)$ et $j = \sigma^{t,1}(i)$.

3.1.1.3 Labelisations

Chaque objet détecté est caractérisé sur chaque image par un ensemble de marqueurs visuels délivrés par l'algorithme de détection. Ces marqueurs permettent d'identifier un objet sur plusieurs images et donc de le suivre. Comme dit précédemment, effectuer un tracking revient à construire une suite de T permutations $(\sigma^t)_{t\leq T}$ où pour tout $t \in \{0, \ldots, T\}$, $\sigma^t \in S_n$, de telle sorte que pour tout $i \leq n$ les marqueurs $A_i^1, A^2_{\sigma^1(i)}, \ldots, A^T_{\sigma^T\circ\ldots\circ\sigma^1(i)}$ soient "proches".

Généralement, le suivi revient à résoudre un problème d'optimisation de la forme :

$$\sigma^{t+1} = \underset{u\in S_n}{\text{Arg inf}} \sum_{i=1}^{n} d\left(A^{t+1}_{u(i)}, F\left(A_i^t, A^{t-1}_{\sigma^{-t}(i)}, \ldots, A^0_{\sigma^{-1}\circ\ldots\circ\sigma^{-(t-1)}\circ\sigma^{-t}(i)}\right)\right) \quad (3.2)$$

F est ici un prédicteur des marqueurs des objets à $t+1$ à partir des marqueurs de ces mêmes objets aux images précédentes. La fonction d mesure la distance ou la dissimilarité qu'il peut y avoir entre les marqueurs à l'instant $t+1$ et ces marqueurs estimés.

3.1.1.4 Connaissance a priori des objets

Il existe deux types de suivi d'objets :

— Premièrement, le suivi d'objet fait suite à une étape de détection (**Tracking by detection**). Chaque objet est défini par une position Z_i^t et une bounding box R_i^t qui retournent la position et la taille de l'objet détecté. Un système Tracking by detection nécessite donc une étape préalable de détection des objets cibles dans toute l'image, à partir de laquelle l'association entre pistes connues et cibles détectées est alors réalisable (BREITENSTEIN

et al., 2009) (BOCHINSKI, EISELEIN et SIKORA, 2017). À noter que les performances des méthodes de tracking par détection sont fortement liées à celles du modèle de détection.

— Deuxièmement, le suivi n'est pas précédé d'une étape de détection (**Détection free tracking**). Dans ce cas une étape d'initialisation est tout de même obligatoire. Elle consiste à marquer d'une bounding box chaque objet présent dans l'image initiale. Le suivi consiste alors à trouver dans les images suivantes les zones de l'image correspondant le plus possible avec les bounding boxes (FRAGKIADAKI et SHI, 2011) (LIN et al., 2015). L'inconvénient de ce type de méthode est qu'elle ne gère pas les entrées de nouveaux objets dans le champ de la caméra. Si un objet n'a pas été initialisé en début de vidéo, celui-ci ne peut donc pas être suivi. En revanche, là où une méthode de tracking par détection est limitée au type d'objet à suivre, une méthode free tracking est indépendante de la nature de l'objet à suivre.

Méthodes d'appariement L'appariement entre une piste connue et un nouvel objet cible détecté à labéliser peut-être obtenu soit par une méthode déterministe soit par une méthode probabiliste.

— **Méthodes déterministes**. Ces méthodes recherchent l'appariement optimal entre pistes connues et objets cibles détectés en se basant sur une mesure de distance entre les marqueurs notés A_i^t dans la section 3.1.1.3. Ces méthodes diffèrent entre elles par le type de marqueurs caractérisant chaque objet (section 3.1.2), par le type de métrique utilisée pour mesurer la distance entre ces marqueurs (section 3.1.3) et par la façon de résoudre le problème d'optimisation (Équation 3.2). La façon la plus commune d'aborder ce problème, c'est d'estimer a priori la position des cibles en se basant sur les positions de chaque piste. À ce niveau-là plusieurs suppositions sont faites sur le mouvement des objets (Figure 3.2). Les suppositions les plus courantes sont : continuité des vitesses et conservation de l'orientation de déplacement (SETHI et JAIN, 1987). Une fois la position des cibles prédites, s'ensuit la résolution d'un problème d'optimisation combinatoire (Equation 3.2). Chaque objet cible doit être affecté à la piste dont il est le plus proche tout en minimisant la somme des distances de chaque piste avec sa cible. Cette optimisation peut être résolue par des algorithmes gloutons (*greedy algorithm* en anglais). Ceux-ci ont l'avantage d'être rapides à l'exécution, mais sont généralement spécifiques au problème étudié (SINGH, RAJAN et MAJUMDAR, 2017) (PIRSIAVASH, RAMANAN et FOWLKES, 2011). D'autres algorithmes comme celui de Kuhn-Munkres ou algorithme Hongrois (MUNKRES, 1957) fonctionnent pour des cas plus généraux et permettent de résoudre le problème d'affectation en temps polynomial.

— **Méthodes Probabilistes**. Les mesures de détections peuvent être perturbées par du bruit. Ce bruit peut être dû à la faible qualité de l'image, au mouvement de la caméra ou aux limites du modèle de détection. Pour pallier ce bruit, il est possible d'introduire une incertitude sur les mesures.

Ces méthodes sont généralement basées sur l'utilisation du filtre de Kalman (LI et al., 2010) (WENG, KUO et TU, 2006). Parmi toutes ces méthodes, deux sont plus largement utilisées. Il s'agit de **M**ultiple **H**ypothesis **T**racking (**MHT**) (REID, 1979) et **J**oin **P**robabilistic **D**ata **A**ssociation **F**ilter (**JPDAF**) (FORTMANN, BAR-SHALOM et SCHEFFE, 1980) (REZATOFIGHI et al., 2015) . La méthode MHT permet l'appariement d'une piste avec plusieurs objets détectés. Chaque appariement est lié à une probabilité basée sur la distance entre la piste et l'objet cible. Finalement, la piste retenue parmi toutes celles possibles est celle ayant la plus forte probabilité. Une telle méthode a l'avantage d'être peu sensible au bruit de détection, en revanche elle demande une grande puissance de calcul et une grande quantité de mémoire. Pour pallier les problèmes de mémoire, les pistes aux probabilités les plus faibles sont régulièrement supprimées. La méthode **P**robabilistic **D**ata **A**ssociation autorise l'appariement d'une piste avec plusieurs cibles, dans la mesure où celles-ci se trouvent dans une zone "proche" de la position estimée de la nouvelle cible appelée *zone porte* (cercles rouges sur la Figure 3.2). En plus de l'appariement avec chaque objet cible inclus dans la zone porte, la méthode PDA fait intervenir la possibilité qu'aucun appariement ne soit établi. Chaque appariement est pondéré par un facteur lié à la distance entre la cible et la position estimée a priori de celle-ci. Là où la méthode PDA est appliquée à une seule piste, la méthode JPDAF permet de gérer les conflits qu'il peut y avoir entre plusieurs pistes, notamment si celles-ci ont leurs zones portes qui se recoupent et possèdent donc des objets cibles en commun.

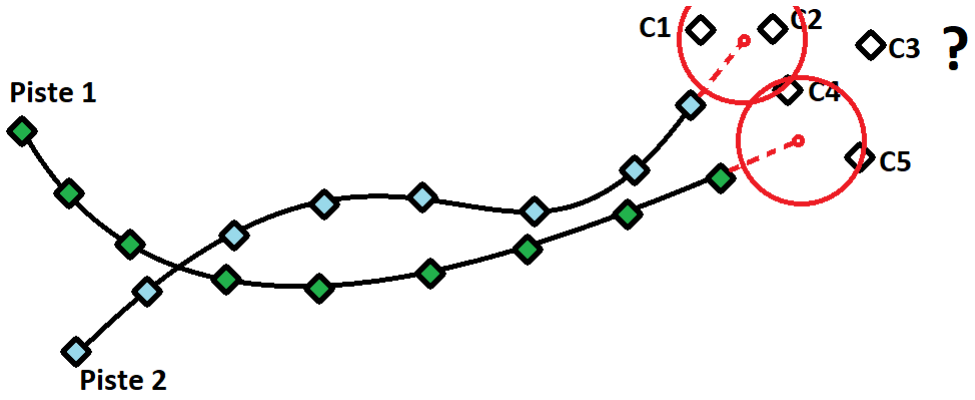

FIGURE 3.2 – Recherche d'une cible pour chaque piste

3.1.1.5 Modes de traitement

Il existe deux modes de traitement qui sont le mode online et le mode offline. La différence entre les deux modes porte sur l'information disponible par l'algorithme de suivi à chaque instant.

— **Le mode online**. À chaque instant le modèle de suivi doit déterminer les meilleurs appariements entre les pistes composées des détections passées

et des cibles représentées par les détections présentes. On utilise le mode online lorsque le tracking doit être effectué au rythme de l'arrivée des images de la vidéo à traiter. On retrouve ce mode de traitement pour de l'analyse de vidéo surveillance afin d'être capable d'envoyer une alerte en temps réel.

— **Le mode offline**. On retrouve le mode offline avec des vidéos préenregistrées. C'est-à-dire que le travail de détection peut avoir été fait en amont du suivi. Dans une telle situation, le modèle de suivi a accès non seulement aux pistes passées, mais également à tous les objets cibles des instants futures. Tester toutes les combinaisons d'association d'objets possibles demanderait des puissances de calcul énormes. Cela n'est actuellement pas possible d'un point de vue pratique à moins de ne suivre qu'un nombre très limité d'objets durant de courts instants. On retrouve donc soit des méthodes qui testent plusieurs possibilités d'association de données, mais durant un court instant (REID, 1979), soit des méthodes qui ne traitent qu'un seul objet à la fois (BERCLAZ, FLEURET et FUA, 2006) (YAN et al., 2006). Une dernière catégorie de méthodes de suivi classée en mode offline se déroule en deux étapes (SONG et al., 2010) (BRENDEL, AMER et TODOROVIC, 2011). Tout d'abord, une première association de données considérées comme évidentes est opérée sur les objets détectés créant ainsi plusieurs pistes de suivi. Cette étape correspond finalement à un mode online. Vient ensuite une étape de recoupement des pistes appartenant à un même objet.

3.1.2 Classification des méthodes de suivi

D'après (YILMAZ, JAVED et SHAH, 2006), il existe trois familles d'algorithme de suivi qui se différencient par la représentation des objets suivis. On retrouve les méthodes de suivi de noyau, celles de silhouettes et enfin de suivi de points. Les deux premières méthodes fonctionnent généralement avec des modèles de suivi *detection free*, alors que l'on retrouve plutôt les méthodes de suivi de points avec des modèles *Tracking by Detection*.

3.1.2.1 Suivi de noyau

Le suivi de noyau, ou également appelé *Kernel Tracking*, analyse l'apparence des objets inclus dans une région bien limitée (généralement rectangulaire ou elliptique). On appelle cette région associée à un objet détecté *région cible*. Chaque objet est représenté par une imagette de référence le représentant que l'on appelle ici *template*. L'idée est alors de trouver dans la nouvelle image des régions cibles qui ressemblent autant que possible aux *templates* de référence. À chaque image, la méthode de suivi estime le mouvement de chaque objet. Puis, elle recherche pour chaque objet la zone de l'image (correspondant à la région cible) autour de la position estimée qui est la plus *"proche"* du *template* de référence. Le suivi est donc lié à la détection. À noter que si le système

de suivi de noyau est utilisé en mode *tracking by detection* une simple correspondance est faite avec les images issues des bounding boxes (3.1.1.1) et les *templates*. La méthode la plus brute se nomme *template matching*. Une corrélation est mesurée entre une imagette de l'image I centrée en (d_x, d_y) et une imagette *template* T de référence de l'objet suivi. Plus le score est élevé, plus la probabilité que ces imagettes appartiennent au même objet est grande.

$$score = ArgMax_{d_x,d_y} \frac{\sum_x \sum_y (T(x,y) * I(x+d_x, y+d_y))}{\sqrt{\sum_x \sum_y T^2(x,y)}}$$

Afin d'être robuste à des changements de luminosité, cette corrélation peut être appliquée sur le gradient de l'image plutôt que sur l'image initiale (BIRCHFIELD, 1998). De façon plus générale l'analyse peut se porter sur n'importe quelle information incluse dans la zone de l'imagette. Cela peut être un histogramme de couleurs (COMANICIU, RAMESH et MEER, 2003) comme n'importe quel vecteur de caractéristiques (ref : section 2.1.1.3). La mesure de similitude entre deux imagettes ou deux descripteurs est présentée dans la section . Une méthode suivie de noyau s'applique généralement à des objets dont la physionomie ne varie pas au cours du temps (ex : une voiture). Cette méthode peut être limitée par les occlusions ou par la sortie progressive de l'objet de sa région de définition, ce qui implique que l'information incluse dans la région cible évolue et s'éloigne de sa référence. D'un autre côté, l'objet peut être caché durant une longue période. Lorsque celui-ci réapparaît, il est alors très facile de l'affecter à sa piste correspondante. Cette méthode, lorsqu'elle analyse des histogrammes, est également indépendante aux rotations de l'objet, ce qui n'est pas le cas de la méthode suivante adaptée aux objets dont l'apparence évolue.
La méthode de flux optique analyse les vecteurs de déplacement de chaque pixel afin d'estimer la position de l'objet dans l'image suivante. Quelques suppositions sont faites sur l'image :
— La luminosité d'une petite région de l'image est constante
— Les pixels voisins appartiennent à un même objet
— Les pixels correspondants à un objet varient graduellement
Cette méthode a été développée par (LUCAS, KANADE et al., 1981), et permet de suivre un objet en analysant le mouvement local de chaque pixel.

3.1.2.2 Suivi de silhouette

Comme dit précédemment, le suivi de silhouette est une méthode plus adaptée à des objets dont la forme évolue au cours du temps. Le suivi de silhouette cherche à modéliser la forme de l'objet et met à jour cette silhouette à chaque image de la vidéo (HUTTENLOCHER, NOH et RUCKLIDGE, 1993) (BERTALMIO, SAPIRO et RANDALL, 2000). Ce ne sont plus des zones cibles qui sont analysées, mais des morceaux d'images aux formes non fixes. Le principal travail des méthodes de suivi de silhouette est un travail de segmentation et de détection de contours (ref : section 2.1.2.1). Tout comme pour le suivi de noyau,

une mesure de similitude est effectuée entre un objet cible et une référence. Cette fois-ci les références correspondent aux objets détectés à l'instant $t-1$ et évoluent donc à chaque instant de la vidéo.
Une silhouette correspond donc à une zone de l'image appelée imagette. La mise en correspondance peut donc être établie entre une imagette cible et une imagette de référence. Dans ce cas on retrouve les correspondances entre histogrammes comme pour le suivi de noyau. Ces silhouettes peuvent également être représentées par le contour de ces imagettes (VADDI et al., 2011). Il n'est donc plus question d'analyser des intensités de pixels (au moyen d'histogrammes), mais des formes d'imagettes. Ces méthodes sont adaptées au suivi d'objets dont l'apparence évolue au cours du temps, grâce à une mise à jour de l'imagette référence. Elles sont robustes à l'évolution de l'apparence des objets suivis, en revanche elles sont sensibles aux occlusions.

3.1.2.3 Suivi de points

Dans le cas des suivis de points, les méthodes se divisent en deux familles. Soit un objet est représenté par un unique point (sa position) soit par plusieurs points (positions de points d'intérêts). Ces points sont généralement issus d'un modèle de détection ce qui place le suivi de points dans la catégorie *tracking by detection*. Les mises en correspondance entre un même objet d'une image à l'autre se fait selon un calcul de distance entre ses positions successives (VEENMAN, REINDERS et BACKER, 2001) (INTILLE, DAVIS et BOBICK, 1997) (SALARI et SETHI, 1990). Si l'on se trouve dans un cas où la trajectoire de l'objet est relativement lisse, on peut donc avoir une idée a priori sur la position future de l'objet (REID, 1979). Ces méthodes de suivi doivent être capables de filtrer les faux positifs issus du modèle de détection et de gérer les occlusions. En effet, si l'objet est représenté par un seul point, la détection fonctionne en tout ou rien. Les méthodes se différencient par le choix de la métrique utilisée pour mesurer la distance entre deux points ou deux groupes de points d'une image à l'autre, et par le choix de la méthode d'appariement qui permet de relier deux points basés sur leur distance (ref : section 3.1.1.4). On utilise une méthode de suivi de points lorsque les objets sont de très petites tailles ou bien que ceux-ci sont visuellement très similaires, empêchant ainsi de les différencier en se basant sur leur apparence.

3.1.3 Métrique

Selon la représentation des objets suivis (voir section précédente), une métrique doit être définie afin de déterminer la similitude qui existe entre deux objets à des temps différents. Plus cette similitude entre deux détections est grande, plus la probabilité que ces détections soient faites sur le même objet est grande. Ces métriques sont représentées soit par des distances, une distance nulle impliquant une grande probabilité d'appartenance au même objet, soit par un indice de similarité compris entre 0 et 1, auquel cas un indice

nul représente cette fois-ci une faible probabilité d'appartenance au même objet.

Représentation contours La mesure de distance entre deux contours correspond alors à la distance minimale entre deux groupes de points (ref section 3.1.2.2). De cette façon Huttenlocher et al. (HUTTENLOCHER, NOH et RUCKLIDGE, 1993) ont développé une méthode basée sur la distance de Hausdorf :

$$D_g(P,Q) = \underset{g\in G}{minmax}\left[h\left(g(P),Q\right),h\left(Q,g(P)\right)\right]$$

avec

$$h(P,Q) = \underset{p\in P}{max}\underset{q\in Q}{min}|p-q|$$

Cette distance mesure la proximité des points entre les deux groupes. G correspond ici à un ensemble de transformations. S'il existe $g \in G$ tel que $g(P) = Q$ alors $D_g(P,Q) = 0$. Ce genre de méthode s'applique généralement lorsque le contour de l'objet varie très peu d'une image à l'autre. En revanche il n'y pas de condition de proximité spatiale.

Représentation par imagettes Cette corrélation est celle mentionnée dans la section 3.1.2.1. Il s'agit de trouver la zone de l'image I^{t+1} à l'instant $t+1$ qui maximise la corrélation avec l'objet *template* O.

$$arg\ max_{d_x,d_y}\frac{\sum_x\sum_y\left(O(x,y)*I^{t+1}(x+d_x,y+d_y)\right)}{\sqrt{\left(\sum_x\sum_y O^2(x,y)\right)}}$$

d_x et d_y correspondent au déplacement de l'objet dans l'image $t+1$.

Représentation par *bounding boxes* R_i^t Dans le cas où un objet détecté est représenté par la région circonscrite à ce même objet, l'indice de Jaccard permet de mesurer la similarité en taille et en position de deux régions (incluses dans ces *bounding boxes*). Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019) définit un indice de dissimilarité entre deux régions A et B comme :

$$IoU(A,B) = 1 - \frac{A\cap B}{A\cup B}$$

Avec :

$$A\cap B = max\left(0;D\right)$$

$$D = \left[min\left(A_{y_{min}},B_{y_{min}}\right) - max\left(A_{y_{max}},B_{y_{max}}\right)\right]\left[min\left(A_{x_{min}},B_{x_{min}}\right) - max\left(A_{x_{max}},B_{x_{max}}\right)\right]$$

Représentation par histogrammes Hwann-Tzong (CHEN et LIU, 2001) utilise la divergence de Kullback-Leibler comme indice de dissimilarité. Il définit tout d'abord une distribution de probabilité des couleurs comme :

$$p(u;X) = \frac{1}{C_p} * \sum_{y_i \in R(X)} w_i(y_i;X)\delta(b(y_i) - u) \tag{3.3}$$

C_p est un coefficient de normalisation, $b(y_i)$ représente la valeur de pixel y_i dans l'histogramme des couleurs, w_i est une fonction de poids permettant de jouer sur la contribution de y_i et où $R(X)$ est la région circonscrite à l'objet centré en X. (CHEN et LIU, 2001) détermine alors la distance entre deux distributions p et q à partir de la distance de Kullback Leibler comme :

$$D(p(u)||q(u,X)) = \sum_{u=1}^{n} p(u) log \frac{p(u)}{q(u;X)}$$

X correspond à la position de l'objet cible dont on cherche à mesurer la similarité. Cette méthode nécessite une apparition non nulle de toutes les couleurs. Une opération de lissage sur l'histogramme des couleurs peut être effectué au préalable. Cette méthode reste tout de même très largement utilisée. Une autre façon de mesurer la similitude entre deux histogrammes est utilisée par Comaniciu (COMANICIU, RAMESH et MEER, 2003), qui calcule la distance de Bhattacharyya entre deux histogrammes. On garde ici la même définition de la distribution $p(u;X)$ utilisée précédemment. La distance est alors écrite comme :

$$\rho(p(u)||q(u,X)) = \sum_{u=1}^{m} \sqrt{q(u;X)p(u)} \tag{3.4}$$

Représentation par des points Lorsque sont connus les positions des objets et les instants auxquels ils ont été détectés, il est alors possible d'avoir une information sur la direction de déplacement, sur la vitesse, voire sur l'accélération de l'objet (Figure 3.4).
Sethi and Jain (SETHI et JAIN, 1987) ont défini des distances normalisées qui privilégient l'uniformité des directions ($d1^t_{i,j}$) et des vitesses ($d2^t_{i,j}$), entre les vecteurs $\mathbf{V}^t_i$ et $\mathbf{V}^{t+1}_k$.

$$d1^t_{i,j} = 1 - \frac{\langle \mathbf{V}^t_i, \mathbf{V}^{t+1}_j \rangle}{\|\mathbf{V}^t_i\| \left\|\mathbf{V}^{t+1}_j\right\|} = 1 - cos\left(\mathbf{V}^t_i, \mathbf{V}^{t+1}_j\right) \tag{3.5}$$

$d1_{i,j}$ est appelée cohérence directionnelle. Le scalaire $\langle \mathbf{V}^t_i, \mathbf{V}^{t+1}_j \rangle$ est le produit scalaire ente $\mathbf{V}^{t+1}_j$ et $\mathbf{V}^t_i$. Puisque $\langle \mathbf{U}, \mathbf{V} \rangle = \|\mathbf{U}\| \|\mathbf{V}\| * cos(\mathbf{U}, \mathbf{V})$, cette cohérence directionnelle peut être vue comme le cosinus entre les vecteurs vitesses $\mathbf{V}^t_i$ et $\mathbf{V}^{t+1}_j$. Cette distance est minimale lorsque les deux vecteurs sont colinéaires et dirigés dans le même sens. Elle est maximale lorsque les vecteurs sont colinéaires, mais de sens opposés.

$$d2_{i,j}^{t} = 1 - \frac{2 * \sqrt{\left\|\mathbf{V}_i^t\right\| \left\|\mathbf{V}_j^{t+1}\right\|}}{\left\|\mathbf{V}_i^t\right\| + \left\|\mathbf{V}_j^{t+1}\right\|} = \frac{\left(\sqrt{\left\|\mathbf{V}_i^t\right\|} - \sqrt{\left\|\mathbf{V}_j^{t+1}\right\|}\right)^2}{\left\|\mathbf{V}_i^t\right\| + \left\|\mathbf{V}_j^{t+1}\right\|} \tag{3.6}$$

$d2_{i,j}$ est appelée cohérence de vitesse. Elle mesure la variation d'amplitude entre les deux vecteurs $\mathbf{V}_i^t$ et $\mathbf{V}_j^{t+1}$. Cette distance est nulle lorsque les magnitudes des deux vecteurs sont égales et augmente lorsque leur différence augmente (Figure B.4). La valeur maximale de la cohérence de vitesse est de 1, elle est atteinte si $\left\|\mathbf{V}_j^{t+1}\right\|$ ou $\left\|\mathbf{V}_i^t\right\|$ est égale à 0, ou si $\left\|\mathbf{V}_j^{t+1}\right\|$ ou $\left\|\mathbf{V}_i^t\right\|$ est infinie.

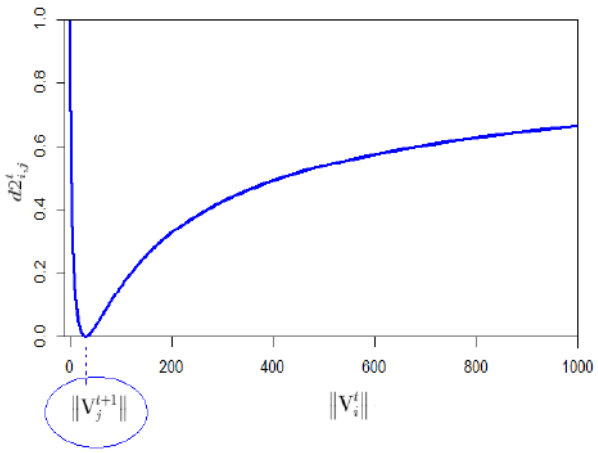

FIGURE 3.3 – Courbe correspondant à la cohérence de vitesse pour une valeur fixe de $\left\|\mathbf{V}_j^{t+1}\right\|$. Ici $\left\|\mathbf{V}_j^{t+1}\right\| = 30$

Cette cohérence de vitesse n'implique pas que la différence entre les deux vitesses soit minimale. Par exemple, la valeur absolue de la différence de vitesse entre $\left\|\mathbf{V}_j^{t+1}\right\| = 30$ et une vitesse $\left\|\mathbf{V}_i^t\right\| = 0$ est plus petite que $\left\|\mathbf{V}_j^{t+1}\right\| = 30$ et $\left\|\mathbf{V}_i^t\right\| = 1000$. Mais d'après la Figure B.4, la cohérence de vitesse $d2_{i,j}^t$ est égale à 1 dans le premier cas, là où elle ne vaut que 0.66 dans le second. En d'autres mots, si un objet se déplace à faible vitesse, cette cohérence de vitesse peut privilégier l'association de cet objet avec un autre objet bien plus loin dans l'image suivante plutôt qu'avec ce même objet. Cette situation apparaît en particulier lorsque l'objet démarre ou s'arrête. Sethi et Jain définissent ensuite $\Delta = w1 * d1_{i,j}^t + w2 * d2_{i,j}^t$ qui est une combinaison des deux vitesses. Les poids w1 et w2 sont choisis entre 0 et 1, de telle sorte que $w1 + w2 = 1$.
La distance définie par Rangarajan (RANGARAJAN et SHAH, 1991) est composée de deux termes. Le premier mesure l'écart entre les vecteurs vitesses formées aux instants t et $t+1$ (même notation que pour Sethi and Jain (SETHI et JAIN, 1987)). Le second terme correspond à l'intensité des vecteurs vitesses

au temps $t+1$.

$$d3^t_{i,j} = \frac{\|Z^{t-1}_{\sigma^{-t}(i)} Z^t_i - Z^t_i Z^{t+1}_j\|}{\sum_{k=1}^{N^t} \sum_{h=1}^{N^{t+1}} \|Z^{t-1}_{\sigma^{-t}(k)} Z^t_k - Z^t_k Z^{t+1}_h\|} + \frac{\|Z^t_i Z^{t+1}_j\|}{\sum_{k=1}^{N^t} \sum_{h=1}^{N^{t+1}} \|Z^t_k Z^{t+1}_h\|}$$

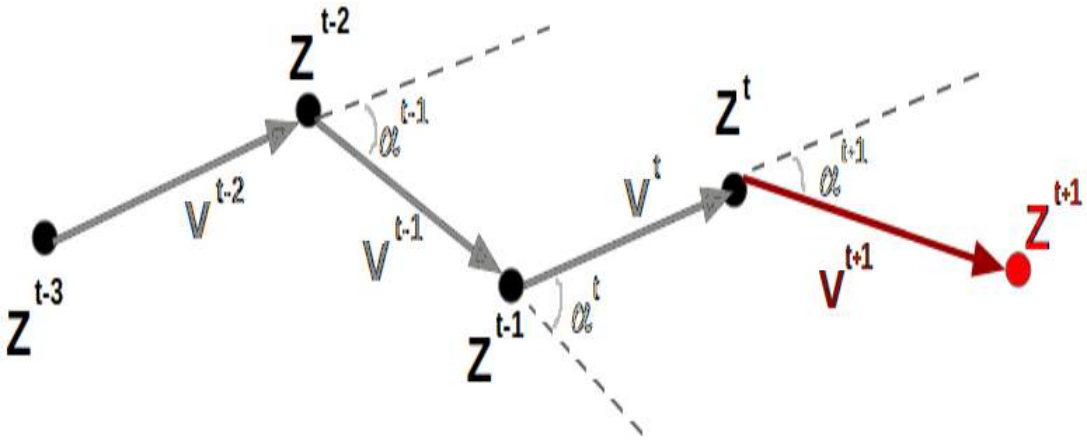

FIGURE 3.4 – Nomenclature pour le suivi de points

3.1.4 Modélisation des déplacements

Plutôt que de mesurer une distance entre une piste et un objet détecté (objet cible), il peut être judicieux d'estimer une distance entre l'objet cible et sa position estimée. Si l'on prend pour exemple la Figure 3.5, celle-ci représente le tracking de deux objets dont les détections passées Z^t_i sont décrites par la piste 1 et la piste 2. Connaissant la position des objets détectés passée, il est alors possible d'estimer les futures positions $\hat{P}_1$ et $\hat{P}_2$ respectivement les pistes 1 et 2. Les objets cibles sont ici représentés par Z_a et Z_b. On peut remarquer sur cette même figure que l'objet cible Z_a est globalement à la même distance du dernier objet détecté de chaque piste qui sont Z^{t-1}_1 et Z^{t-1}_2. En revanche si l'on vient à le comparer aux positions estimées de chaque objet cible, sa distance à la position estimée $\hat{P}_1$ apparaît bien plus faible que sa distance à la position estimée $\hat{P}_2$. Même si cette méthode ne résout pas tous les problèmes d'incertitudes d'affectations, elle contribue tout de même à les simplifier. Cette méthode repose sur la façon dont sont estimées les futures positions. Meilleures sont les estimations, plus courtes sont les distances entre positions mesurées et positions estimées. De courtes distances facilitent alors l'appariement des points. Ce chapitre est tourné principalement sur la meilleure façon d'estimer le mouvement des poulets de chair. S'il est possible de trouver un modèle qui modélise au mieux le mouvement des poulets, cela diminue alors le risque de mauvaises affectations.
Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019) définit une modélisation du déplacement. La distance est alors égale à zéro si la différence entre la position prédite et la position estimée est nulle. La distance est égale à un si

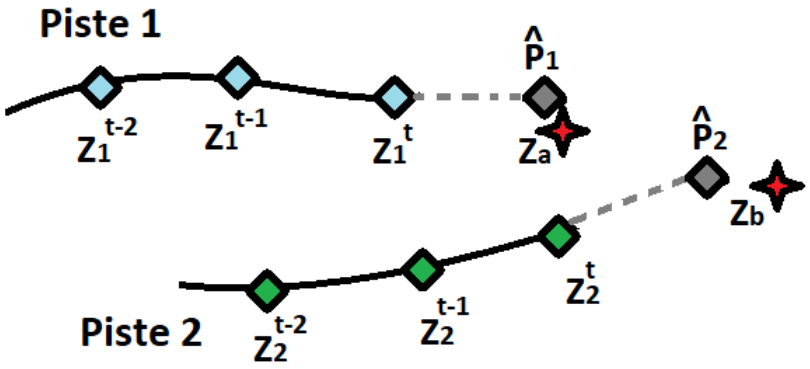

FIGURE 3.5 – Estimation de la position

cette différence est supérieure à la taille de l'objet. Ce déplacement maximum de l'objet est écrit n_{dst}. Il est déduit des dimensions de la région circonscrite à l'objet, issue de la détection (R_i^t). Cela sous-entend qu'entre deux images le déplacement de l'objet est inférieur aux dimensions de sa taille. La définition de la distance entre la position de l'objet à la position $Z_{\sigma_i^{t+1}}^{t+1}$ et sa position estimée $\hat{Z}_{\sigma_i^{t+1}}^{t+1}$ est écrite comme :

$$d4 = min\left(1, \frac{|Z^{t+1} - \hat{Z}^{t+1}|}{n_{dst}}\right)$$

Avec $\hat{Z}^{t+1}$ définie comme une moyenne mobile pondérée des M instants précédents :

$$\hat{Z}t + 1 = Z^t + \frac{\sum_{a=1}^{M-1} a\left(Z^{t+a-M+1} - Z^{t+a-M}\right)}{n_{dst}}$$

À noter que le choix des coefficients de pondération du calcul de la moyenne mobile est arbitraire.
Fletcher (FLETCHER, WARWICK et MITCHELL, 1991) définit une méthode qui n'est pas sensible à un choix de coefficients comme l'est la méthode précédente. Cette méthode est basée sur les séries temporelles et peut être exprimée en trois équations :

$$\hat{Z}_{\sigma^1(i)}^{t+1} = \mathbf{a}(t) * Z_i^{-T_n}$$

$$\mathbf{E} = Z_{\sigma^1(i)}^{t+1} - \hat{Z}_{\sigma^1(i)}^{t+1}$$

$$\mathbf{a}(t) = \mathbf{a}(t-1) + \mathbf{K} * \mathbf{E}$$

Avec :
— $Z_i^{-T_n}$ les n positions passées de Z_i^t
— $\mathbf{K}$ un vecteur de n constantes
— $\mathbf{a}(t)$ le vecteur de coefficients fixé automatiquement
La position estimée est calculée en multipliant les positions passées avec un ensemble de coefficients mis à jour automatiquement. Cette mise à jour est opérée en tenant compte de l'erreur de prédiction *a posteriori*. Cette méthode a

l'avantage d'être adaptée au mouvement des objets suivis. Chaque objet à son propre ensemble de coefficients $\mathbf{a}(t)$. Cependant, elle nécessite une méthode indépendante pour initialiser le vecteur $\mathbf{a}(t)$.
Les suppositions faites par Sethi et Jain (SETHI et JAIN, 1987) sur l'uniformité du déplacement (orientation et vitesse constantes) peuvent également être adaptées à de la modélisation de mouvement comme par exemple :

$$\hat{Z}^{t+1} = Z^t + V^t$$

3.2 Mise en œuvre de la méthode de suivi

3.2.1 Définitions et objectifs

Au cours du tracking nous avons à disposition un ensemble de poulets détectés aux temps passés à qui un identifiant a déjà été attribué. Chaque poulet est identifié par une position, une bounding box et un identifiant. L'ensemble des positions passées d'un poulet s'appelle une piste. Pour chaque point composant une piste, le plus ancien (premier détecté) correspond au début de piste et le plus récent (dernier détecté) correspond à la fin de piste. Ce dernier point de piste est appelé *point référence*. Chaque piste est associée à un identifiant unique. Une piste est dite *active* si son point référence correspond à un poulet toujours dans le champ de la caméra. La piste est dite *morte* si son point référence correspond à un animal sorti du champ de la caméra. Ce point référence se situe donc sur une extrémité de l'image et plus aucune affectation ne peut lui être proposée. À l'instant *t+1*, une nouvelle image passe à travers le réseau de neurones qui fournit alors un ensemble de position $Z^{t+1}_{i \leq N_{t+1}}$, N_{t+1} étant le nombre de poulets détectés à l'instant $t+1$. Ces points Z^{t+1}_i sont appelés *points cibles*. L'objectif est d'associer une piste à chaque poulet détecté à l'instant *t+1*. C'est la position des poulets issue du réseau de neurones qui permet une telle association. Globalement un point cible Z^{t+1}_i est associé à la piste dont le point référence (Z^t_j) est le plus proche selon la distance L2 (distance Euclidienne). Le choix de la distance L2 s'explique par le fait que celle-ci est invariante pour toute rotation de l'image. Les distances entre poulets restent inchangées par rotation de caméra. Le tracking doit cependant faire face à de nombreuses contraintes.

— Chaque poulet n'est pas forcément représenté par un mais par plusieurs points si plusieurs détections sont faites sur la même image (voire section 2.3.2.4).
— Il n'y a pas forcément autant de pistes actives que de nouveaux poulets détectés. Certains peuvent entrer ou sortir du champ de la caméra.
— Tous les poulets ne sont pas forcément détectés. Le réseau de neurones n'est pas parfait, et bien qu'appliquer plusieurs détections améliore le nombre de poulets détectés, il en reste toujours qui peuvent ne pas être détectés sur certaines images.

— Les poulets étant en mobiles, il est possible d'avoir plusieurs poulets détectés dont la piste la plus proche est la même. De la même façon plusieurs pistes peuvent être proches d'un même poulet détecté à l'instant t.
— S'il est possible d'avoir plusieurs points pour un même poulet dû au fait que plusieurs détections soient appliquées sur l'image, il est également possible d'avoir plusieurs points pour un même poulet sur une seule et même détection. Ce peut être le cas lorsqu'un poulet passe sous l'abreuvoir, auquel cas le poulet peut être détecté en deux parties (une de chaque côté de l'abreuvoir). C'est très souvent le cas également lorsqu'un poulet bats des ailes.
— De la même façon qu'un poulet peut correspondre à plusieurs détections, plusieurs poulets peuvent correspondre à une seule et même détection. C'est le cas lorsqu'un poulet en cache un autre ou que ceux-ci sont extrêmement serrés.
— Si le réseau de neurones peut rater la détection de certains poulets, il est également possible que celui-ci génère des faux-positifs. Il génère alors des positions et de bounding box de poulets à des endroits de l'image sans poulets.
— Toutes affectations entre une piste et un poulet détecté ne sont pas possibles. Il existe certaines limites physiques dont il faut tenir compte. En un dixième de seconde un poulet ne traverse pas le champ de la caméra qui fait plusieurs mètres de large. L'affectation d'un poulet avec une liste trop éloignée ne doit pas être acceptée.

La figure 3.6 reprend l'ensemble de ces obstacles qui peuvent créer des contraintes au bon déroulement du tracking
Le tracking peut alors se divises en plusieurs étapes :

1. Fixer les seuils. Cette opération s'effectue en tout de début de tracking. Elle permet de fixer les distances maximales de déplacement (d_{max}) et les distances minimales (d_{min}) autorisées entre deux poulets.
2. Supprimer le surplus de points. Puisque chaque animal peut être représenté par plusieurs points, il est alors nécessaire de ne garder qu'un seul point par poulet avant toute affectation.
3. Estimer pour chaque piste la position du nouveau point. Plutôt que de chercher à appareiller le dernier enregistrement d'une piste avec un point issu du réseau de neurones, il est préférable de représenter une piste par un point qui est une estimation de la position du nouveau point détecté (voire section 3.1.4). Cette étape est expliquée plus amplement par la suite et a fait l'objet d'un papier (voire Appendix B).
4. Appareiller chaque point détecté avec un point estimé. Cette opération est effectuée au moyen d'un algorithme d'optimisation combinatoire appelé algorithme Hongrois (KUHN et TUCKER, 2014). Chaque appariement entre un point détecté et un point estimé à un coût basé sur la distance entre ces points. L'algorithme Hongrois retourne l'appariement optimal des points qui garantisse un coût total minimum.

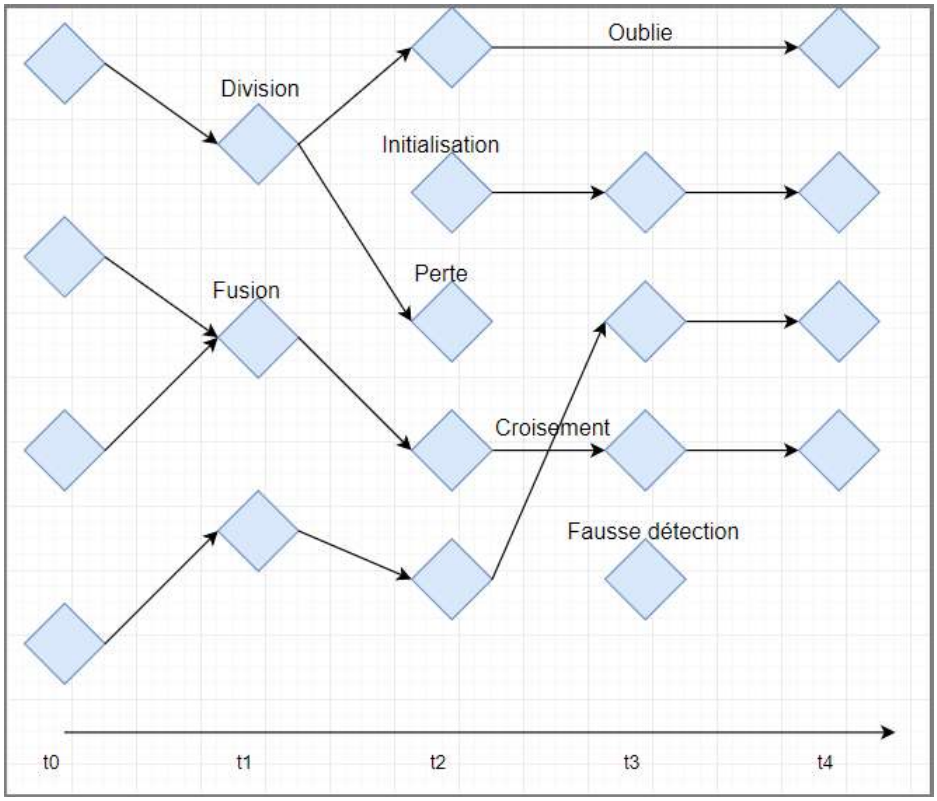

FIGURE 3.6 – Résumé des évènements qui impliquent des complications de tracking

5. Corriger les appariements. Comme dit précédemment, ce tracking doit faire face à de nombreuses contraintes. Cette étape permet de vérifier que toutes les contraintes soient bien respectées.
6. Enregistrer les données. Une fois les appariements vérifiés, chaque point est alors associé à une piste et à l'identifiant correspondant. Les nouveaux points n'ayant pas été affectés à une piste sont alors considérés comme des nouveaux poulets entrant dans le champ de la caméra à qui un nouvel identifiant est attribué. Ils correspondent donc au début de nouvelles pistes.

3.2.2 Étapes du tracking

3.2.2.1 Estimation des seuils

Les deux seuils qui doivent être fixé sont : la distance minimale entre deux poulets et la distance maximale de déplacement.

Le seuil de distance minimale : Deux points dont la distance est inférieure à ce seuil minimal sont considérés comme appartenant au même poulet. Si ce seuil est trop faible, deux points risques de correspondre au même poulet. Un seuil trop élevé supprime alors des points lorsque des poulets sont trop proches.

Le seuil de déplacement maximal : Tout point Z_i^{t+1} dont la distance à un point estimé $\hat{Z}_j^{t+1}$ est supérieure au seuil de déplacement maximal est considéré comme étant un poulet différent. Un seuil trop faible empêche de suivre les poulets qui se déplace trop vite et dont l'estimation de la position n'est pas bien adaptée. Un seuil trop élevé laisse la possibilité d'appareiller un nouveau point détecté à de multiples pistes. Plus ce nombre de pistes est élevé, plus le risque de mauvais appariement est élevé (Figure 3.7).

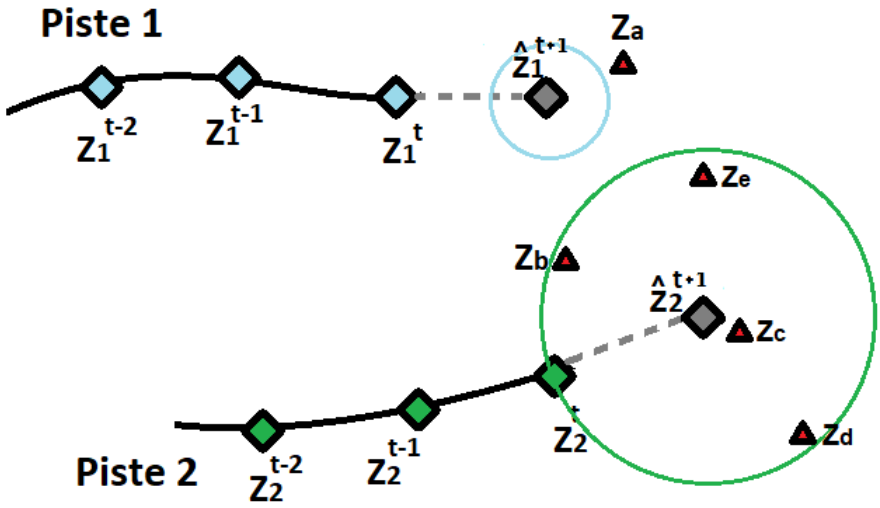

FIGURE 3.7 – La distance de déplacement maximale de la piste 1 est trop petite, le nouveau point détecté Z_a ne peut donc pas lui être affectée. La distance maximale de la piste 2 est trop grande, de trop nombreux points peuvent être affectés à la piste 2, là où seul le point Z_c semble correspondre

L'idéal serait d'avoir des seuils uniques que l'on puisse fixer expérimentalement et qui soient les plus adaptés à toutes situations. Or si l'on prend par exemple le seuil de distance minimale, il est facile de comprendre que la grosseur des poulets joue sur cette distance. Plus les poulets sont gros, plus la distance qui sépare leur position centrale est élevée. Grâce à la détection, il est alors possible d'avoir une information sur la taille des poulets à travers les bounding box. Ces largeurs et hauteur de bounding box servent alors d'étalonnage pour fixer ces seuils.

3.2.2.2 Sélection des points à appareiller

Il est question ici de sélectionner un seul point par poulet parmi tous ceux issus du réseau de neurones. Le choix des points à sélectionner s'appuie d'un part sur le nombre de points voisins à une distance d_{min} et de la plus courte distance entre chaque point à sélectionner et les plus proches fins de piste. Si deux points sont séparés d'une distance inférieure à d_{min}, le point sélectionné pour le tracking est celui dont la distance à une fin de piste la proche est la plus courte. Cette méthode garantie un seul point par poulet quel que soit le nombre de propositions faites par le réseau de neurones.

3.2.2.3 Estimation du mouvement

Comme mentionné dans la section 3.1.4, il est préférable de chercher à associer les nouveaux points détectés avec les points estimés de chaque piste plutôt que directement avec chaque fin de piste. Meilleure est l'estimation du mouvement de l'animal, plus petite est le seuil de distance maximale de déplacement. Avec une fréquence de capture d'image très élevée (par exemple 50 images/secondes) les positions des poulets ne varient que très peu d'une image à l'autre. Avec 50 images/seconde il suffit alors de rechercher la fin de piste la plus proche de chaque nouveau poulet détecté. Le problème d'une fréquence de capture d'image trop élevée, c'est le temps de traitement extrêmement long qui est lié. Un modèle Faster R-CNN est capable d'opérer entre 5 et 20 images par seconde pour travailler en temps réel. Cela ne tient compte que de la détection et varie en fonction de la puissance du GPU utilisé et de la taille des images. Dans notre cas, le modèle Faster R-CNN est capable de traiter une image d'un mégapixels en 58 ms. Sachant que les images traitées font au moins 2M pixels, il n'est pas envisageable de traiter des vidéos à plus de 10 images/secondes. De plus sur de grandes images (2 à 3M pixels), la détection fonctionne en fenêtre glissante pour couvrir toute l'image. Cela augmente le nombre de détections et devient très chronophage. Enfin, si la qualité de la détection n'est pas assez bonne pour l'utilisation souhaitée, il peut être nécessaire d'appliquer plusieurs détections sur chaque image comme mentionné précédemment. C'est pourquoi 10 images/seconde est actuellement la plus haute fréquence de captation d'image envisageable. A 10 images/seconde le mouvement d'objets telle qu'une voiture varie très faiblement de vitesse et d'orientation de mouvement. C'est pourquoi en se basant seulement sur les 2 derniers points d'une piste, le vecteur mouvement a estimé $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$ doit être sensiblement égal au dernier vecteur mouvement enregistré $\mathbf{V}^{t}_{i}$. À 10/images par seconde, cette supposition de mouvement régulier n'est pas forcément une bonne façon d'estimer un mouvement de poulet. Le mouvement d'un poulet est bien moins régulier que celui d'une voiture. C'est pourquoi, une analyse a été faite afin de déterminer le meilleur modèle qui soit capable de bien estimer le mouvement de poulets. Ce modèle doit pouvoir estimer le mouvement d'un poulet en se basant seulement sur ces vecteurs mouvements précédents. Il doit également être robuste aux différentes erreurs de détection. La détection n'étant pas parfaite, des poulets peuvent être manqués sur certaines images.

Pour cette étude plusieurs modèles ont été testés, certain venant de la littérature et d'autres ont été créés pour le projet. Actuellement, il n'existe pas dans la littérature un modèle de modélisation de mouvement appliqué spécialement au poulet de chair. L'idée est de tester les différents modèles déjà existants, voir s'ils sont adaptés au poulet de chair et d'en proposer des nouveaux.

3.2.2.3.a Les modèles

Les différents modèles de prédiction de mouvement sont les suivants :

Supposition de mouvement nul. Ce modèle est lié à la notion de plus proche voisin. C'est un modèle de référence qui part du principe que le mouvement des animaux est nul. C'est un modèle bien adapté à des animaux qui ne bougent pas.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \epsilon^t_i \tag{3.7}$$

Avec : $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$.

Mouvement constant. Ce modèle est tiré de l'article de Sethi et Jain (SETHI et JAIN, 1987). Ce modèle suppose que le mouvement de l'animal reste constant au cours du temps.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \mathbf{V}^t_i + \epsilon^t_i \tag{3.8}$$

Avec : $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$.

Moyenne pondérée des vecteurs mouvements passés. Ce modèle est tiré de l'article de Hasith Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019). Le vecteur mouvement $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$ est une combinaison linéaire des P vecteurs mouvements passés.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \gamma_0 \mathbf{V}^t_i + \sum_{s=1}^{P} \gamma_s \mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} + \epsilon^t_i \tag{3.9}$$

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \frac{1}{3}\mathbf{V}^t_i + \frac{1}{15}\sum_{s=1}^{4}(5-s)\mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} + \epsilon^t_i \tag{3.10}$$

Avec $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$

Combinaison de vitesses et variation d'orientation. Ce modèle tient compte des P précédentes vitesses et variations d'orientation de façon indépendante. La vitesse d'un animal dépend donc de ses vitesses passées. Son orientation de déplacement dépend de ses variations d'orientation passées. La Figure 3.8 représente les différentes nomenclatures utilisées pour l'élaboration de ce modèle. Dans un souci de simplicité, le vecteur déplacement $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$ est ici exprimé en coordonnées polaires.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \left(R^{t+1}_{\sigma^{t+1}(i)}, \Theta^{t+1}_{\sigma^{t+1}(i)}\right) \tag{3.11}$$

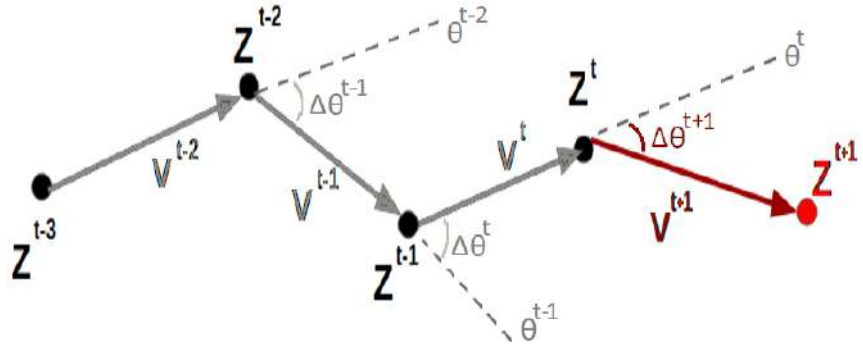

FIGURE 3.8 – nomenclature

Avec

$$\Theta^{t+1}_{\sigma^{t+1}(i)} = \Delta\Theta^{t+1}_i + \Theta^t_i$$

$$R^{t+1}_{\sigma^{t+1}(i)} = \left(R^t_i\right)^{\alpha_0} * \prod_{s=1}^{P} \left(R^{t-s}_{\sigma^{-(t-s+1),-t}(i)}\right)^{\alpha_s} * \exp^{\epsilon_{R^t_i}}$$

$$\Delta\Theta^{t+1}_{\sigma^{t+1}(i)} = \beta_0 \Delta\Theta^t_i + \sum_{s=1}^{P} \beta_s \Delta\Theta^{t-s}_{\sigma^{-(t-s+1),-t}(i)} + \epsilon_{\Theta^t_i}$$

Avec $\epsilon_{Ri} \sim \mathcal{N}(0, \Sigma_R)$ et $\epsilon_{\Theta i} \sim \mathcal{N}(0, \Sigma_\Theta)$

3.2.2.3.b Utilisation des modèles

Pour ce faire, une base de données a été créé grâce à trois vidéos d'une minute (10 images/seconde) avec entre 34 et 68 poulets en moyenne par image selon les vidéos. Pour chaque vidéo la position de tous les animaux a été relevée à la main. Un premier tracking a été effectué au moyen de cette base de données de référence. Après vérification de la justesse du tracking (pas d'erreur d'identification), une base de données d'animaux trackés (chacun avec un unique identifiant) est alors disponible afin d'estimer les paramètres de certains modèles. L'avantage est d'avoir des modèles dont les paramètres sont calibrés spécialement sur un poulet de chair.

Plus proche voisin (PPV). Cette première méthode estime le mouvement en se basant sur le modèle B.12. Ce modèle ne nécessite pas de paramètres à estimer.

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = 0 \tag{3.12}$$

Mouvement constant (MC). Cette deuxième méthode d'estimation de déplacement se base dur le modèle B.15. Là aussi, le modèle ne nécessite pas l'estimation de paramètres.

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \mathbf{V}^t_i \tag{3.13}$$

Moyenne pondérée (MP). Cette troisième méthode est basée sur le modèle B.13. Tout comme dans l'article de Hasith Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019) (d'où provient ce modèle), les paramètres γ_s et P sont fixés. Le paramètre P est fixé à 5. Les paramètres γ_s appartiennent à l'ensemble : $1, 2, 3, 4, 5$. Le total étant ensuite normalisé par la somme des paramètres γ_s.

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \frac{1}{3}\mathbf{V}^t_i + \frac{1}{15}\sum_{s=1}^{4}(5-s)\mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} \tag{3.14}$$

Moyenne pondérée estimée (MME). Cette quatrième méthode est également basée sur le modèle B.13. Cette fois-ci les paramètres ne sont pas fixés, mais estimés. Les paramètres γ sont estimés au moyen de la base de données de référence par méthode des moindres carrés. Le paramètre P est fixé de telle façon que la contribution de chaque terme $\gamma_s \mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)}$ soit supérieur à un demi-pixel. Grâce à la base de données de référence, il est alors possible de connaître la vitesse maximale qu'atteint un poulet. Globalement les vecteurs mouvements sont associés à des paramètres qui décroissent avec le temps. Plus le vecteur mouvement correspond à un temps lointain, plus le paramètre associé est petit. Dès lors que ces paramètres associés à la vitesse maximale enregistrée apportent une contribution inférieure à un demi pixel, ces temps passés ne sont alors pas pris en compte. Un premier test a été effectué avec une valeur de P à 10, afin de déterminer la valeur de P à retenir. L'estimation du vecteur mouvement est alors écrit :

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \gamma_0 \mathbf{V}^t_i + \sum_{s=1}^{P}\gamma_s \mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} \tag{3.15}$$

Chaque vecteur mouvement $\mathbf{V}^t_i$ possède 2 composantes : $\mathbf{V}^t_i = (\mathbf{V}_{X^t_i}, \mathbf{V}_{Y^t_i})$. Pour des raisons d'invariante par rotation, il est nécessaire d'avoir les mêmes paramètres pour les composantes X et Y. L'estimation des paramètres est donc ici réalisée dans le domaine complexe, $\mathbf{V}^t_i = \mathbf{V}_{X^t_i} + i\ \mathbf{V}_{Y^t_i}$. La résolution des moindres carrés dans le domaine complexe est donnée par Turetsky (TURETSKY, 1951).

On note $A = B + iC \in \mathcal{R}^{m,n}$ la matrice d'entrée représentant les vecteurs mouvements passés.

$$A = B + iC = \begin{pmatrix} \mathbf{V}_{X_1^t} & \cdots & \mathbf{V}_{X^{t-n}_{\sigma^{-(t-n+1),-t}(1)}} \\ \vdots & & \vdots \\ \mathbf{V}_{X_m^t} & \cdots & \mathbf{V}_{X^{t-n}_{\sigma^{-(t-n+1),-t}(m)}} \end{pmatrix} + i \begin{pmatrix} \mathbf{V}_{Y_1^t} & \cdots & \mathbf{V}_{Y^{t-n}_{\sigma^{-(t-n+1),-t}(1)}} \\ \vdots & & \vdots \\ \mathbf{V}_{Y_m^t} & \cdots & \mathbf{V}_{Y^{t-n}_{\sigma^{-(t-n+1),-t}(m)}} \end{pmatrix}$$

$b = u + iv \in \mathcal{R}^m$ représente les vecteurs mouvements au temps $t+1$.

$$b = u + iv = \begin{pmatrix} \mathbf{V}_{X^{t+1}_{\sigma^{t+1}(1)}} \\ \vdots \\ \mathbf{V}_{X^{t+1}_{\sigma^{t+1}(m)}} \end{pmatrix} + i \begin{pmatrix} \mathbf{V}_{Y^{t+1}_{\sigma^{t+1}(1)}} \\ \vdots \\ \mathbf{V}_{Y^{t+1}_{\sigma^{t+1}(m)}} \end{pmatrix}$$

Enfin, $z = x + iy \in \mathcal{R}^n$ est le vecteur des paramètres γ à estimer. Seule la partie réelle est retenue au final.

$$z = x + iy = Re \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{pmatrix} + i\ Im \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{pmatrix}$$

La solution est écrite sous la forme :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} BB^T + CC^T & -C^TB + B^TC \\ C^TB - B^TC & BB^T + CC^T \end{pmatrix}^{-1} \begin{pmatrix} B^T & C^T \\ C^T & B^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{3.16}$$

Vitesses et Variations d'orientation (RT). Cette cinquième méthode est basée sur le modèle 3.11. Les paramètres sont également estimés par méthode des moindres carrés. Cette fois-ci les vecteurs mouvements sont estimés en coordonnées polaires. Ces deux composantes sont associées à des paramètres différents (α_s pour la composante R, et β_s pour la composante Thêta). Le paramètre P est estimée de la même manière que pour la méthode (**MME**) en se basant sur la contribution de la composante R des vecteurs mouvements. Un point important doit être pris en compte quant aux valeurs que peuvent prendre les deux composantes R et Thêta. Les valeurs de Thêta sont comprises entre 0 et 2π, de ce fait les valeurs des composantes R doivent être positives.

Un angle peut représenter soit une direction, soit une rotation. Alors qu'il est possible de sommer des rotations (deux rotations de $\pi/4$ équivaut à une rotation de $\pi/2$), sommer des directions n'a pas d'interprétation physique. C'est pourquoi l'estimation de la composante Thêta de $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$ passe par l'estimation $\Delta\Theta^{t+1}_{\sigma^{t+1}(i)}$ qui est elle même une combinaison linéaire des précédentes variations d'orientation. *In fine*, $\Theta^{t+1}_{\sigma^{t+1}(i)} = \Delta\Theta^{t+1}_{\sigma^{t+1}(i)} + \Theta^t_{(i)}$. Les valeurs de Thêta sont ainsi comprise entre $[-\pi, \pi)$. Puisque les valeurs de Thêta

peuvent prendre toutes les valeurs de direction possible, il est alors nécessaire de restreindre les composantes R à des valeurs positives. Passer en logarithme afin d'estimer les paramètres du modèle permet d'assurer des valeurs positives à la composante R de $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$.

$$\begin{pmatrix} log(R_1^{t+1}) \\ \vdots \\ log(R_m^{t+1}) \end{pmatrix} = \begin{pmatrix} log(R_1^{t}) & \dots & log(R_1^{t-n}) \\ \vdots & & \vdots \\ log(R_m^{t}) & \dots & log(R_m^{t-n}) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}$$

Et ainsi :

$$\hat{R}^{t+1}_{\sigma^{t+1}(i)} = \left(R_i^t\right)^{\alpha_0} * \prod_{s=1}^{P} \left(R^{t-s}_{\sigma^{-(t-s+1),-t}(i)}\right)^{\alpha_s}$$

$$\Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)} = \beta_0 \Delta\Theta_i^t + \sum_{s=1}^{P} \beta_s \Delta\Theta^{t-s}_{\sigma^{-(t-s+1),-t}(i)}$$

Vitesses et Variations d'orientation sous contraintes (RTSC). Cette méthode d'estimation des vecteurs déplacements reprend la précédente (**RT**), à laquelle est appliquée quelques contraintes. Globalement, plus un animal se déplace vite, plus ses variations d'orientation de déplacement diminuent. L'idée n'est pas de modéliser parfaitement la relation qu'il existe entre vitesse et variation d'orientation, mais plutôt d'empêcher l'apparition de valeurs aberrantes. Pour de faibles valeurs de vitesses, toutes les variations d'orientation doivent être autorisées. En revanche plus la vitesse augmente, plus les variations d'orientation autorisées sont restreintes. La Figure 3.9 représente l'histogramme en 2 dimensions des couples (R, $\Delta\Theta$) rencontrés dans la base de données de référence. Les droites rouges représentent les limites des couples autorisés.

Une première estimation des vitesses et des variations d'orientation est opérée avec la méthode (**RT**). Puis les grandes vitesses sont acceptées pour de faibles orientations et les grandes variations d'orientation pour de faibles vitesses estimées.

$$\hat{R}^{t+1}_{\sigma^{t+1}(i)} = \left(R_i^t\right)^{\alpha_0} * \prod_{s=1}^{P} \left(R^{t-s}_{\sigma^{-(t-s+1),-t}(i)}\right)^{\alpha_s}$$

$$\Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)} = \beta_0 \Delta\Theta_i^t + \sum_{s=1}^{P} \beta_s \Delta\Theta^{t-s}_{\sigma^{-(t-s+1),-t}(i)}$$

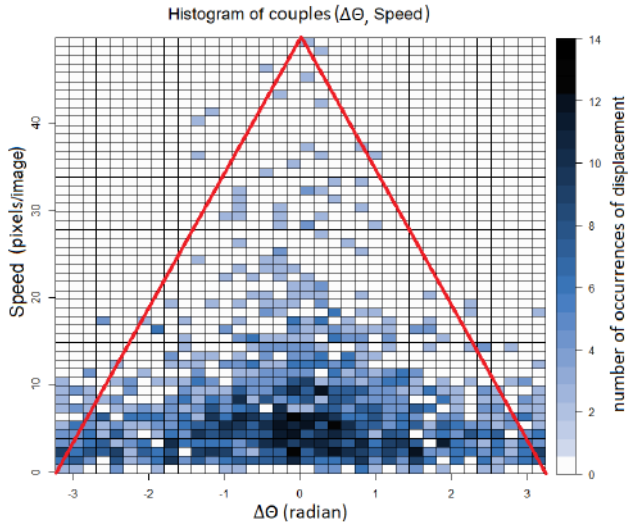

FIGURE 3.9 – Relation entre vitesse et variation d'orientation. Les droites rouges correspondent aux limites des couples (R, ΔΘ) autorisés

Une fois $\hat{R}^{t+1}_{\sigma^{t+1}(i)}$ et $\Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)}$ estimés, ils sont utilisés pour contraindre les nouveaux estimateurs $\hat{R}'^{t+1}_{\sigma^{t+1}(i)}$ et $\Delta\hat{\Theta}'^{t+1}_{\sigma^{t+1}(i)}$

$$\hat{R}'^{t+1}_{\sigma^{t+1}(i)} = \hat{R}^{t+1}_{\sigma^{t+1}(i)} \left(1 - abs\left(\frac{\Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)}}{\pi}\right)\right)$$

$$\Delta\hat{\Theta}'^{t+1}_{\sigma^{t+1}(i)} = min\left(\Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)}, \pi\left(1 - \frac{\hat{R}^{t+1}_{\sigma^{t+1}(i)}}{R_{max}}\right)\right)$$

La valeur R_{max} est mesurée à partir de la vitesse maximale reportée par la base de données de référence.

3.2.2.4 L'algorithme Hongrois

L'algorithme de Kuhn-Munkres (ou algorithme Hongrois) (MUNKRES, 1957) a été choisi comme algorithme d'appariement des points $\mathbf{Z}^{t+1}_{\sigma^{t+1}(i)}$ avec les points $\hat{\mathbf{Z}}^{t+1}_{\sigma^{t+1}(i)}$. C'est un algorithme d'optimisation combinatoire qui résout un problème d'affectation en minimisant un coût total. Notre problème peut être représenté par un graphe biparti dont les sommets sont d'une part les points Z^{t+1} (ensemble A) et d'autre part les points $\hat{Z}^{t+1}$ (ensemble B). Les arêtes du graphe sont représentées par les distances euclidiennes entre ces points. Ces distances sont toutes positives. Le but est de faire matcher les points du groupe A avec ceux du groupe B en sélectionnant les arrêtes du

graphe qui minimisent l'expression :

$$\sum_{a \in A, b \in B} c_{ab} x_{ab}$$

x_{ab} correspond aux valeurs des arêtes, et $c_{ab} \in \{0, 1\}$ selon que le point 'a' est oui ou non affecté au point 'b'. Toutes les valeurs des distances sont représentées par une matrice de coût $M = (x_{a,b})$. Dans le cas idéal chaque point $Z^{t+1}_{\sigma^{t+1}(i)}$ a son équivalent parmi les points $Z^{t+1}_{\sigma^{t+1}(i)}$, $Card(A) = Card(B) = N$. Il y a donc exactement N affectations et chaque point Z^t est affecté à un unique $\hat{Z}^t$ (et inversement).

$$\sum_{a \in A} c_{ab} = 1 \quad \forall \quad b \in B$$

et

$$\sum_{b \in B} c_{ab} = 1 \quad \forall \quad a \in A$$

L'algorithme fonctionne par ajout et retrait de valeur sur chaque ligne et chaque colonne de la matrice de coût. À chaque opération, l'affectation optimale du résultat de la matrice de coût est la même que pour la matrice d'origine. L'idée, c'est de faire apparaître au moins un zéro par ligne et par colonne de la matrice. L'emplacement de ces zéros associe les éléments des lignes 'a' avec les éléments 'b' des colonnes de la matrice. Lorsque le nombre de points varie au cours du tracking (disparition, fusion ...), on risque d'avoir à faire à de fausses affectations. Il faut donc limiter ce risque de fausses affectations d'une part et identifier celles qui doivent être autorisées d'autre part. Dans le cas de l'algorithme de Kuhn-Munkres si une affectation est fausse, celle-ci peut alors engendrer toute une série de fausses affectations (Figure 3.10 et Table 3.1).

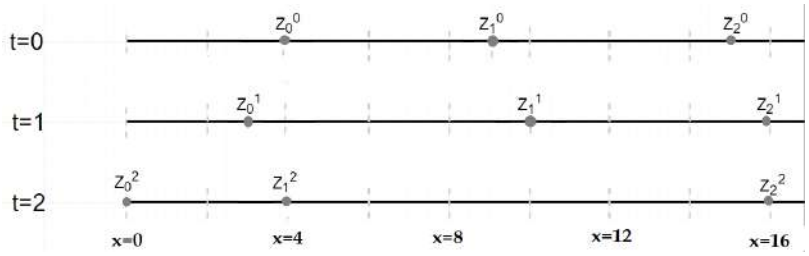

FIGURE 3.10 – Exemple d'application de l'algorithme Hongrois. Cet exemple est associé au tableau 3.1 qui présente les différentes associations de données de cet exemple

Le premier tableau correspond à l'affectation des points entre les temps t0 et t1, et le deuxième tableau aux temps t1 et t2. La première affectation semble relativement claire, chaque point à t0 ayant un point proche à t1. Par soucis de simplicité, il n'est pas question d'estimation de mouvement pour cet exemple. La somme des distances est alors de 1+1+1=3, c'est le minimum

	Z_0^0	Z_1^0	Z_2^0
Z_0^1	1	6	12
Z_1^1	6	1	5
Z_2^1	12	7	1

	Z_0^1	Z_1^1	Z_2^1
Z_0^2	3	10	16
Z_1^2	1	6	12
Z_2^2	13	6	0

TABLE 3.1 – Exemple d'association de données par l'algorithme Hongrois. Le tableau de gauche représente les affectations entre le temps t=0 et le temps t=1. Le tableau de droite représente les affectations entre le temps t=1 et le temps t=2. Les données sont tirées de l'exemple de la Figure 3.10

	${Z_0}^1$	${Z_1}^1$	${Z_2}^1$
Z_0^2	3	10	16
Z_1^2	1	6	12
Z_2^2	13	6	0

	${Z_0}^1$	${Z_1}^1$	${Z_2}^1$
Z_0^2	3	10	10
Z_1^2	1	10	10
Z_2^2	10	10	0

TABLE 3.2 – Exemple d'association de données par l'algorithme Hongrois entre le temps t=1 et le temps t=2. Le tableau de gauche correspond à cette association de données avant l'application des contraintes de distance et le tableau de droite après l'application des contraintes. Les données sont toujours tirées de l'exemple de la Figure 3.10

que l'on puisse avoir. Pour la deuxième affectation on peut considérer deux cas. Soit les points Z_0^1 et Z_1^1 ont subi une translation de leur position, auquel cas leur affectation au temps 2 avec les points Z_0^2 et Z_1^2 est correcte pour une somme des distances de 3+6+0=9 (somme minimale). Soit, on considère que le point Z_1 à disparu à l'instant t2 et que le point Z_0^2 est un nouvel élément, auquel cas cette affectation de l'algorithme de Kuhn-Munkres ne nous satisfait pas.

On essaie ici d'affecter des positions de poulets entre différents temps. On a donc une contrainte de distance qui empêche un poulet de parcourir une trop grande distance entre deux images successives. L'exemple précédent montre que des apparitions et des disparitions de poulets du champ de la caméra peuvent perturber les affectations. Afin de ne plus être sensibles à ces valeurs extrêmes, on force toutes distances supérieures à la distance maximale autorisée d_{max} à une valeur égale à $2 \times d_{max}$. Le facteur 2 est choisi afin d'avoir une différence entre une distance dite 'forcée' et une valeur proche de d_{max}. On reste sur une minimisation totale de la somme des distances, mais ce forçage à $2 \times d_{max}$ privilégie des affectations plus locales. Le but est qu'une distance dite forcée soit la même pour tous les points éloignés (Tableau 3.2).

Dans le cas de figure précédent, Z_0^1 était affecté à Z_0^2, Z_1^1 avec Z_1^2 et Z_2^1 avec Z_2^2.

En fixant une valeur $d_{max} = 5$ (arbitrairement pour cet exemple), on empêche alors une affectation entre les points Z_1^2 et Z_1^1 (voir Table 3.2 de droite). Cette fois-ci la somme des distances est égale $1 + 10 + 0 = 11$. C'est une valeur plus grande que précédemment, mais l'on a permis une affectation plus locale de nos points. Cette fois-ci Z_0^1 et Z_1^2 sont bien affecté ensemble. On ne tient pas compte alors de l'affectation entre Z_1^1 et Z_0^2 car la distance entre ces deux points alors de $2 \times d_{max}$. On considère donc Z_1^1 comme une fin de piste et Z_0^2 comme un début de piste.

3.2.2.5 Correction des affectations

Si un déplacement entre deux images d'un même poulet venait à être supérieur à d_{max}, la correction est alors appliquée lors d'un post-traitement. Il vaut mieux avoir d_{max} trop petit et perdre un poulet qui se déplace rapidement, qu'avoir un d_{max} trop important et voire des erreurs d'identification entre poulets pour cause de mauvaises détections ou de mauvaises estimations de déplacement (voir section 3.2.2.1). Il est plus facile de corriger la perte d'un poulet qui va trop vite en post-traitement qu'un échange d'identifiants entre poulets lors du tracking. Dans le premier cas, l'enregistrement des positions d'un poulet est alors représenté par plusieurs pistes. Le post-traitement vise alors à assembler ces pistes qui appartiennent à un même poulet. Dans le deuxième cas, il est bien plus compliqué de détecter l'appartenance d'une même piste à plusieurs poulets.

Une première façon de limiter les erreurs de suivi dues à une mauvaise détection, est de mémoriser temporairement la position des poulets (dernier point d'une piste) n'ayant pas eu d'affectation avec un point cible à l'instant $t + 1$. Avoir une période de mémorisation trop faible peut ne pas laisser le temps de détecter le poulet à nouveau, surtout lorsque le taux de détection est faible. Un temps de mémorisation trop important augmente le risque de mauvaise affectation, si un poulet venait à entrer dans la zone proche de la position mémorisée d'un poulet perdu momentanément. Ce temps de mémorisation est estimé expérimentalement au vu des résultats de détection.

Un dernier choix porte sur la fusion et la division de poulets. Pour rappel, une fusion correspond à deux poulets représentés par un même point en sortie du réseau de neurones et une division correspond à un unique poulet représenté par deux points. Pour ces deux problèmes, on a à faire à 4 cas de figure présentées Figure 3.11 :
Sur ces 4 cas de figure, deux d'entre eux doivent être autorisés (1 et 3), et les deux autres interdits (2 et 4). Malheureusement il n'est pas si facile de différencier ces situations lorsque tout ce que l'on traite se résume en une liste de positions. Il est alors préférable de privilégier le passage d'une seule détection à deux détections, pour la bonne raison qu'il est plus facile de supprimer en post-traitement l'enregistrement d'un poulet qui n'en est pas un, plutôt que de créer un enregistrement de positions pour un poulet disparu. De plus,

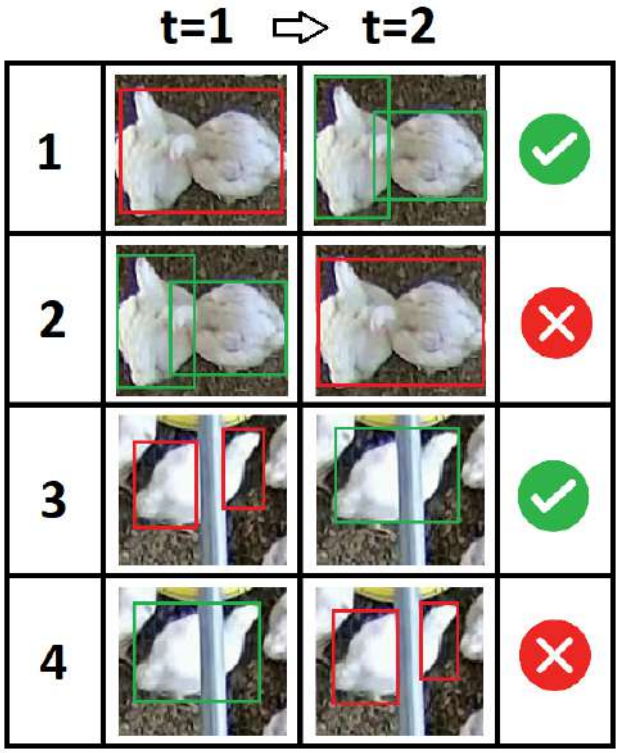

FIGURE 3.11 – Présentation de 4 cas de figure impliquant une détection pour deux animaux ou deux détections pour un seul animal.

si la distance minimale autorisée entre deux poulets a bien été estimée, la configuration du poulet représenté par deux points ne doit avoir qu'une très faible probabilité d'apparaître. *In fine*, il faut être capable d'éviter autant que possible la configuration 2, et savoir attribuer le bon identifiant au poulet pour la configuration 3 (savoir lequel retenir des 2 identifiants références). Afin d'éviter la configuration 2, un point cible n'est pas autorisé à se trouver à une distance inférieure d_{min} de deux point références simultanément. Dans ce cas, le point cible en question est alors supprimé.

Pour la résolution de la configuration 3, apparaît alors la notion de poulets jumeaux. Deux détections avec deux identifiants différents peuvent ici correspondre au même poulet. Le poulet jumeau apparaît surtout lorsqu'un poulet bat des ailes ou passe sous l'abreuvoir et que l'on se retrouve avec plusieurs détections pour un même animal. Il faut donc faire en sorte que lorsque l'animal cesse de battre des ailes ou a traversé la zone abreuvoir, l'identifiant qui lui est attribué soit bien le même qu'avant qu'une double détection apparaisse (Figures 3.12). Pour cela, lorsqu'une piste B est créé près du point référence d'une piste A, cette piste B devient jumelle de la piste A. Si et seulement si, la piste A vient à mourir près du point référence de la piste B. On enregistre alors les identifiants de ces deux pistes qui appartiennent au même poulet, et la correction est alors faite en post-traitement. La figure 3.13 schématise une telle situation. L'algorithme 2 présente la vérification des associations issues de l'algorithme Hongrois.

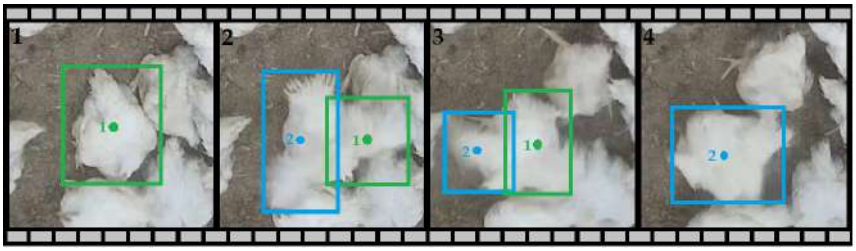

FIGURE 3.12 – Exemple d'erreur de tracking où un poulet est représenté par deux positions

Algorithm 2 Vérification de l'affectation des points cibles

d_{max} = <distance maximale de déplacement>
d_{twin} = <distance maximale à laquelle peut se trouver un poulet jumeau>
for pnt1 in points cibles **do**

if pnt1 a matché avec un point estimé pnt0 **then**

if distance(pnt0, pnt1) > d_{max} **then**
<pnt0 est memorisé>
<pnt1 devient le début d'une nouvelle piste>

if pnt0 avait un point cible à une distance inférieure à d_{twin} **then**
<création d'un point jumeau>
end if
end if
elsepnt1 est affecté à la piste du pnt0 auquel il est associé

if <pnt1 avait un point estimé à une distance inférieure à> d_{twin} **then**
<création d'un point jumeau>
end if

end if

3.2.2.6 Enregistrement des données

A l'issue de cette étape de suivi, les données enregistrées correspondent à une liste de points correspondant à la position de chaque poulet au cours du temps. Chaque point est enregistré avec les informations suivantes :
— **Temps** : Le temps correspond à au numéro de l'image dans la vidéo.
— **Identifiant** : Au cours du suivi chaque point s'est vu attribué un identifiant qui le lie au poulet auquel il correspond. Idéalement (sauf perte

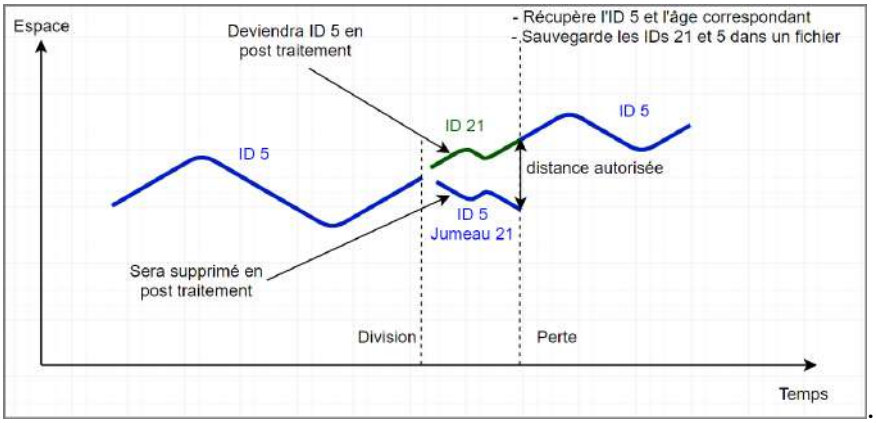

FIGURE 3.13 – Exemple de la gestion du jumeau

du poulet au cours du tracking), chaque poulet est représenté par une unique piste. L'identifiant du point enregistré, correspond à l'identifiant de la piste à laquelle il appartient.

— **X** : C'est la position selon l'axe X du point dans l'image.
— **Y** : C'est la position selon l'axe Y du point dans l'image.
— **L** : C'est la largeur selon l'axe X de la box (issue de la détection) encadrant le point dans l'image.
— **H** : C'est la hauteur selon l'axe Y de la box (issue de la détection) encadrant le point dans l'image.
— **Pixel** : C'est l'intensité du pixel de l'image où se trouve le point enregistré. Un poulet Ross 308 étant globalement de couleur plus claire que la litière, l'analyse de l'intensité de ce pixel permet la reconnaissance de faux positifs (le réseau de neurones a détecté un poulet là où il n'y en avait pas).

Un deuxième jeu de données correspond à la liste des identifiants des pistes *jumelles* et des identifiants des pistes auxquelles elles correspondent. Ce jeu de données est utilisé en post-traitement afin de corriger des erreurs d'identification.

Chapitre 4

Résultats et discussions

Ce chapitre présente tout d'abord les résultats liés à la détection. Ils traduisent la capacité qu'a le réseau de neurones à bien détecter tous les poulets dans le champ de la caméra. Puis sont présentés les résultats du suivi des poulets. Ces résultats traduisent la capacité du système à suivre les poulets le plus longtemps possible sans erreurs. Enfin ce chapitre présente les différentes données qui sont créés à partir des données de tracking.

4.1 Détection

4.1.1 Définitions

Une bonne détection doit à la fois être capable de détecter tous les animaux présents sous le champ de la caméra et ne rien détecter d'autre. Elle ne doit rien détecter d'autre dans la mesure où seules des images de poulets ont été apprises par le réseau de neurones (et non les mangeoires, les abreuvoirs et autres ...). Le tableau 4.1 et la Figure 4.1 représentent ce que l'on appelle des Vrais Positifs, des Faux Positifs et des Faux Négatifs.

— Un Vrai Positif (**VP**) est un animal présent sous le champ de la caméra et bien détecté par le réseau de neurones.
— Un Faux Positif (**FP**) est une détection faite par le réseau de neurones là où aucun poulet n'est présent.
— Un Faux Négatif (**FN**) est un animal qui n'a pas été détecté par le réseau de neurones.

À noter que la notion de Vrai Négatif n'existe pas ici. Un Vrai Négatif serait toute zone de l'image sans poulet où le réseau de neurones n'aurait rien détecté. À partir de ces trois définitions (Vrai Positif, Faux Positif, Faux Négatif), il est alors possible de définir deux types de mesure.

— La sensibilité représente la capacité qu'a le réseau de neurones à bien détecter tous les animaux présents sous le champ de la caméra. Une sensibilité de 100% traduit le fait que tous les animaux présents sont détectés. En revanche une sensibilité de 0% traduit le fait qu'aucun animal présent n'a été détecté. Cette mesure de sensibilité est indépendante du nombre de faux positifs. Elle est définie comme :

$$Sensibilite = \frac{VP}{VP + FN} = \frac{\#\,de\,bonnes\,detections}{\#\,total\,de\,poulets} \tag{4.1}$$

— Le Taux de Fausses Détections traduit la capacité qu'a le réseau de neurones à ne détecter que les poulets présents sous le champ de la caméra. Un taux de 0% correspond à une détection sans aucun Faux Positif. Cette mesure est indépendante du nombre de Faux Négatifs. Il arrive également d'utiliser le terme de Précision qui n'est autre que le complément du taux de fausses détections. Ce taux de Fausses Détections est défini comme :

$$Taux\,de\,Fausses\,Detections = \frac{FP}{VP+FP} = \frac{\#\,de\,fausses\,detections}{\#\,total\,de\,detections} \tag{4.2}$$

$$Precision = \frac{VP}{VP+FP} = 1 - \ taux\,de\,fausses\,detections \tag{4.3}$$

FIGURE 4.1 – Représentation de ce que l'on appelle un Vrai Positif, un Faux Positif et un Faux Négatif

	Poulet présent	Pas de poulet
Détection	Vrai Positif	Faux Positif
Pas de détection	Faux Négatif	–

TABLE 4.1 – Tableau de contingence appliqué à la détection de poulets

4.1.2 Fenêtres de détection

Un modèle Faster R-CNN accepte différentes tailles d'image en entrée. La taille maximale d'image d'entrée acceptée par notre modèle est 1000×1000 pixels. Si la taille des images de la vidéo à analyser est supérieure à 1000×1000 pixels, il faut donc utiliser un système de fenêtres glissantes afin de pouvoir analyser l'ensemble du champ des images de la vidéo. Bien que le modèle ait été entraîné sur des images de poulets de différents âges, celui-ci

détecte plus facilement les poulets plus âgés, et plus difficilement les jeunes poulets. De plus, la base de données qui a servi d'entraînement au réseau de neurones utilise exclusivement des captations réalisées à des hauteurs entre 2 mètres et 2,5 mètres de haut. Par la suite du projet le modèle de détection a été testé sur des images issues de captations réalisées jusqu'à 5 mètres de haut. Encore une fois, les résultats sont bien meilleurs pour des captations réalisées à 2 mètres comparés à ceux des captations réalisées à 5 mètres.

Ces différences dans les résultats peuvent être dues à plusieurs raisons. Dans le premier cas les jeunes poulets sont peut-être sous représentés dans la base de données d'entraînement. Dans le deuxième cas, le réseau de neurones a été entraîné avec une certaine résolution d'image, et élever la caméra réduit alors cette résolution. Le réseau de neurones est limité aux informations qui lui ont été présentées lors de l'entraînement.

Il y a un point commun entre ces deux limites du modèle de détection. Que

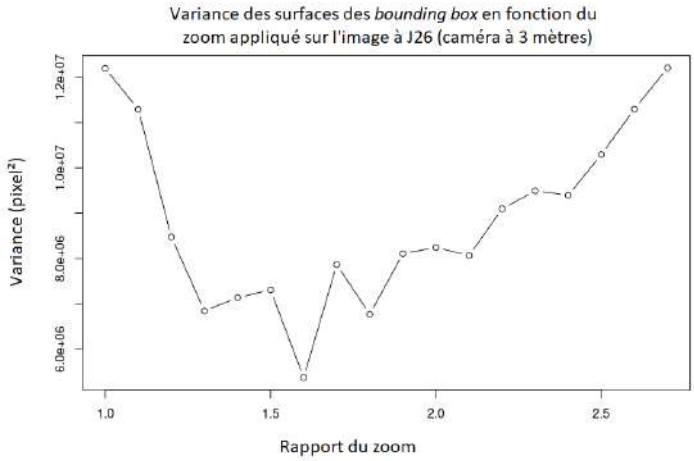

FIGURE 4.2 – Recherche du meilleur zoom à appliquer sur l'image afin d'obtenir une détection optimale

ce soit pour l'âge du poulet ou la hauteur de la caméra, le modèle est limité par la taille apparente de l'animal dans l'image. Puisqu'un modèle Faster R-CNN fonctionne par succession de réduction de l'image d'entrée (voir *pooling* dans la section 2.1.3.2), un objet trop petit dans l'image initiale risque de disparaître à travers ces opérations de *pooling*. Agrandir l'image d'entrée permet de contourner ce problème de définition de l'objet. Si l'image n'est pas assez agrandie, peu d'objets sont alors détectés, et les *bounding box* sont généralement trop grandes pour les objets. Si l'image est trop agrandie, de nombreux faux positifs apparaissent dans l'image. La Figure 4.3 représente la détection de poulets de 26 jours par une caméra 3M pixels placée à 5 mètres de haut selon différents agrandissements de l'image en entrée du réseau de neurones. Cet exemple permet de visualiser l'intérêt d'appliquer la détection sur une taille d'image bien spécifique.

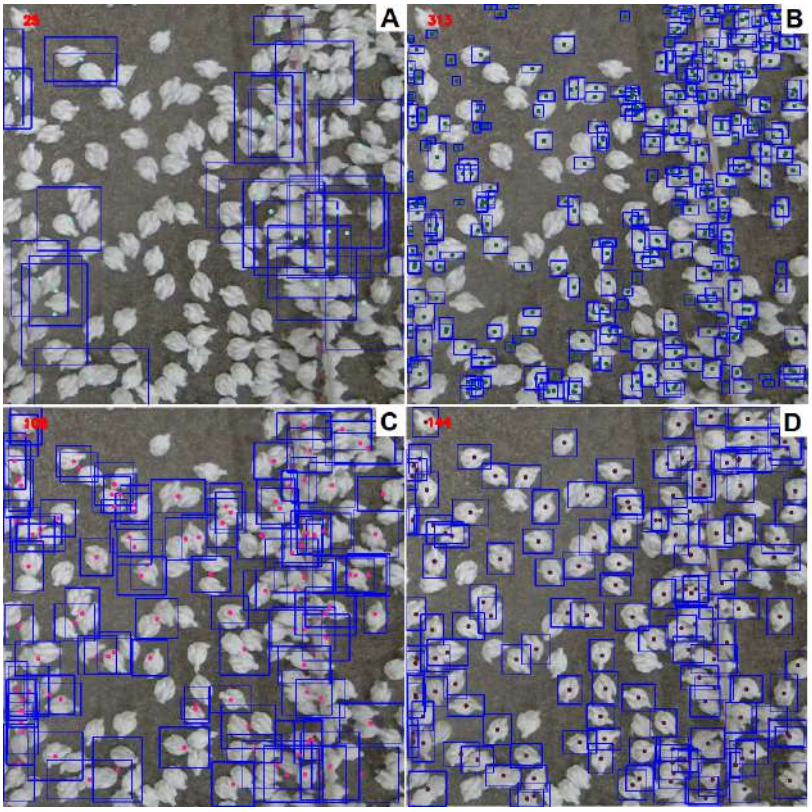

FIGURE 4.3 – Cette figure présente la détection de poulets de 26 jours par une caméra 3M pixels placée à 5 mètres de haut selon différents agrandissements de l'image en entrée du réseau de neurones. Image A : image réduite à 50% de la taille initiale (image trop petite). Image B : image agrandie à 350% de la taille initiale (image trop grande). Image C : image initiale (image trop petite). Image D : image agrandie à 280% de la taille initiale (taille d'image optimale)

Il faut donc pourvoir déterminer le taux d'agrandissement de l'image idéal, et ce, de façon automatique. La métrique qui permet de déterminer un score optimal à partir des mesures de *sensibilité* et de *précision* s'appelle le *score F1*. Il est défini comme :

$$F1 = 2 * \frac{precision * sensibilite}{precision + sensibilite}$$

Ce score permet de déterminer le ratio d'agrandissement de l'image qui retourne le meilleur couple sensibilité - précision. La Figure A 4.4 permet de visualiser l'évolution du score F1 en fonction de la précision et de la sensibilité. Sur cette figure a été tracée un exemple de l'ensemble des couples sensibilité-précision obtenu pour différents zooms appliqués à l'image. La figure B représente cette l'évolution du score F1 en fonction du zoom appliqué sur l'image. Le point A correspond à un zoom pour lequel la plupart des poulets détectés correspondent à des faux positifs. La précision est donc proche

de 0. Le point B correspond à un zoom pour lequel tous les poulets sont détectés et pour lequel il n'y a pas de faux positifs. La précision et la sensibilité sont alors à leur maximum et le score F1 est donc égale à 1. Enfin le point C correspond à un zoom pour lequel il y a autant de vrais positifs que de faux positifs, mais également un zoom pour lequel peu de vrais positifs ont été détectés. La précision est donc proche de 0,5 et la sensibilité proche de 0. La figure C permet de lire les valeurs de précision et de sensibilité en fonction du zoom que l'on ne retrouve pas dans la figure B. Finalement l'information comprise dans la figure A est répartie entre les figures B et C.

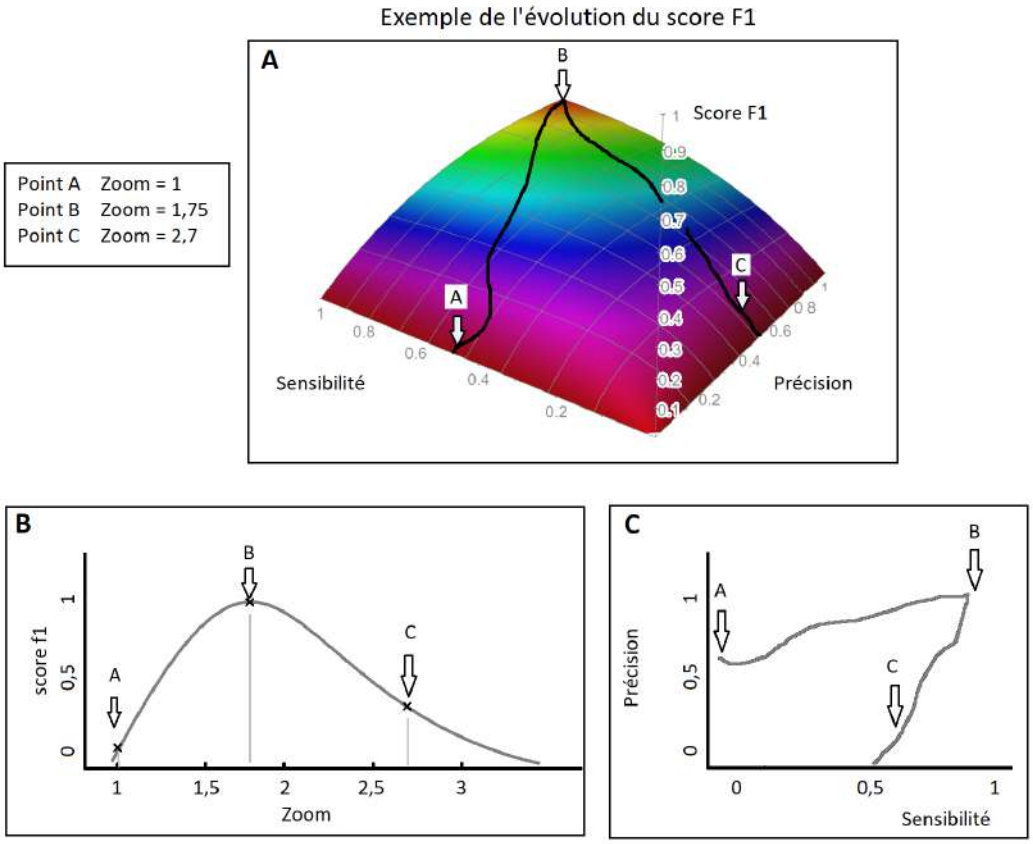

FIGURE 4.4 – La figure A représente l'évolution du score F1 en fonction de la précision et de la sensibilité. La figure B est un exemple de l'évolution du score F1 en fonction du zoom appliqué sur l'image. LA figure C représente l'évolution de la sensibilité et de la précision en fonction des zooms appliqués à l'image. Les points A, B et C permettent de faire le lien entre les deux figures.

La Figure 4.5 permet de visualiser le meilleur ratio à appliquer à chaque image en fonction de l'âge des poulets et de la hauteur de la caméra. La Figure 4.6 permet de visualiser les scores de sensibilité et de précision liés aux scores f1 de la Figure 4.5. La réelle position des poulets a été relevée expérimental ment afin de pouvoir définir les Vrais Positifs, les Faux Positifs et les Faux Négatifs. En pratique ces informations ne sont pas disponibles lors d'un usage

non expérimental du modèle de détection. Il faut donc définir une métrique qui se rapproche le plus possible des scores F1 mesurés précédemment.

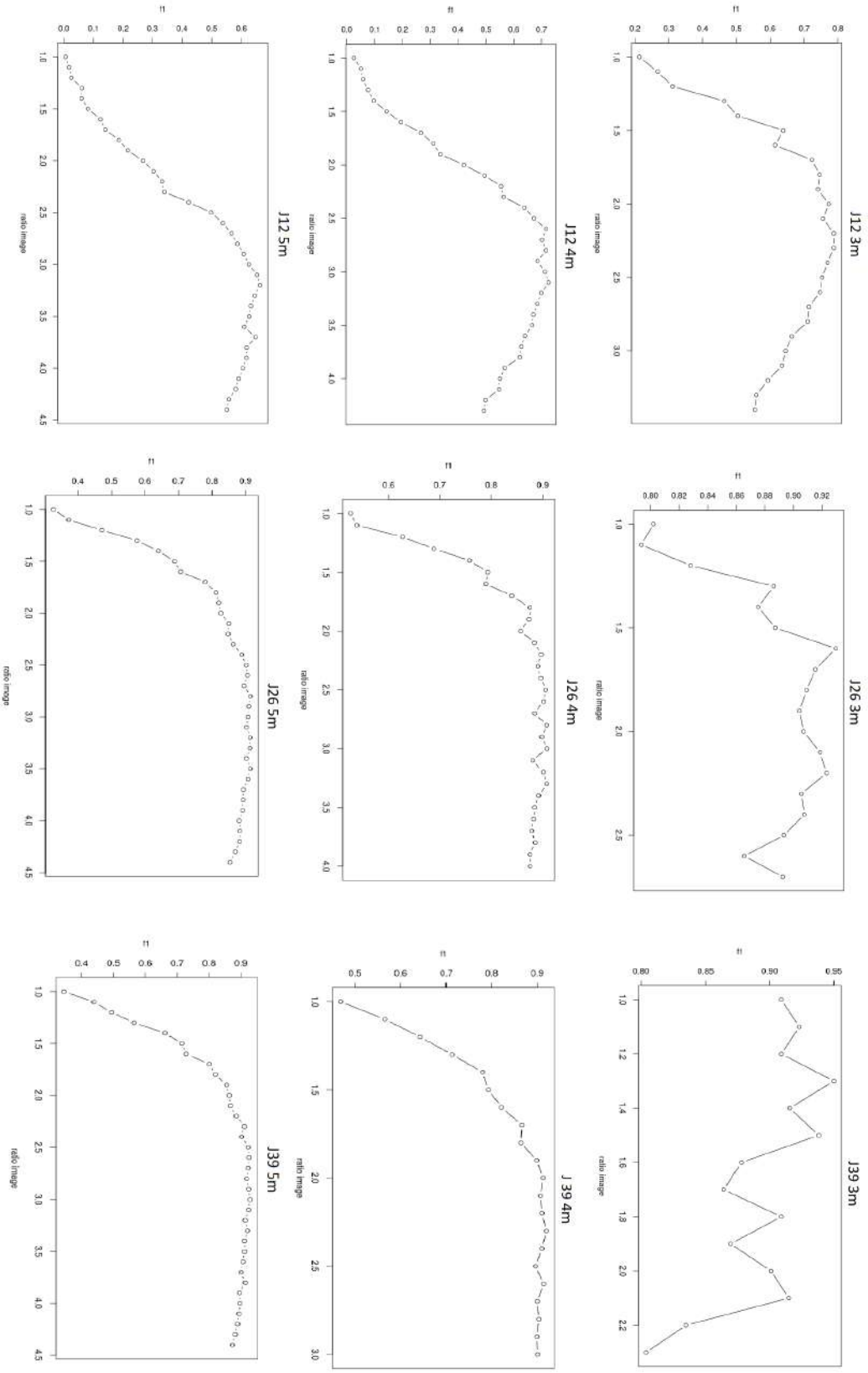

FIGURE 4.5 – Présentation du score F1 selon différents agrandissements de l'image en fonction de l'âge des poulets et de la hauteur de la caméra.

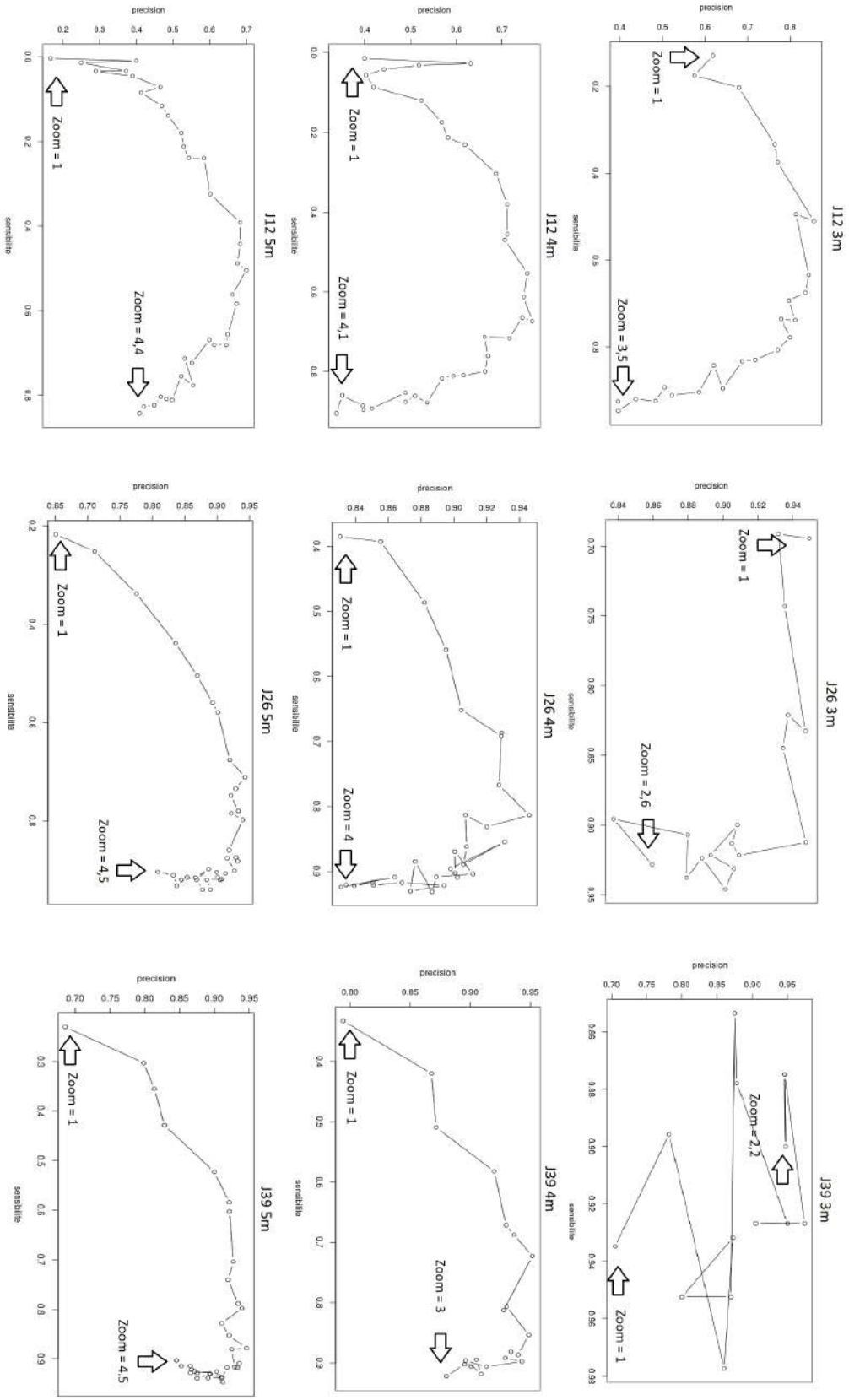

FIGURE 4.6 – Présentation des courbes sensibilité - précision selon différents agrandissements de l'image en fonction de l'âge des poulets et de la hauteur de la caméra.

	J12	J26	J39
2,8 mètres	2,7	1,6	1,2
	2,2	**1,6**	**1,3**
3,7 mètres	3,4	2,5	2,3
	3,1	**3,0**	**2,3**
5 mètres	3,5	3,2	2,3
	3,2	**2,8**	**2,6**

TABLE 4.2 – Tableau retournant le ratio d'agrandissement optimal d'après l'analyse de variance des surfaces. Les valeurs en gras correspondent aux ratios retournés par l'analyse du score f1.

	J12	J26	J39
2,8 mètres	2,3	1,4	1,7
	2,2	**1,6**	**1,3**
3,7 mètres	3,4	2,2	2,1
	3,1	**3,0**	**2,3**
5 mètres	3,5	2,9	2,2
	3,2	**2,8**	**2,6**

TABLE 4.3 – Tableau retournant le ratio d'agrandissement optimal d'après l'analyse de la différence entre le 15^{eme} et le 85^{eme} quantile des surfaces mesurées. Les valeurs en gras correspondent aux ratios retournés par l'analyse du score f1.

Dans la mesure où les poulets ont globalement tous la même taille, des mesures de surfaces qui s'écarteraient de la moyenne ont alors de grandes probabilités d'appartenir à de Faux Positifs. Tracer la variance des *bounding box* en fonction du coefficient d'agrandissement appliqué à l'image permet de déterminer l'agrandissement optimal à appliquer aux images de la vidéo. La Figure 4.2 est un exemple de courbes représentant la variance des *bounding box* en fonction du coefficient d'agrandissement. Dans le cas d'une caméra de $3M$ pixels placés à 3 mètres de haut et pour des poulets âgés de 26 jours, l'agrandissement qui retourne une variance des surfaces minimale est donc de 160% par rapport à la taille de l'image initiale. Dans ce cas particulier, c'est le même agrandissement que celui retourné par l'analyse du score F1 (Voir Figure 4.5). Une deuxième métrique possible mesure la différence de surface entre le 15^{eme} et le 85^{eme} quantile des surfaces mesurées. Encore une fois, en se basant sur le fait que la plupart des poulets ont la même taille, la différence entre ces deux quantiles est d'autant plus faible qu'il y a peu de Faux Positifs dans l'image.
Globalement, la recherche du meilleur ratio basé sur l'analyse de la variance a tendance à surestimer ce ratio pour les jeunes poulets et à sous-estimer ce ratio pour les poulets plus âgés. Le ratio retourné par la méthode de différence de surface entre le 15^{eme} et le 85^{eme} quantile est légèrement inférieur à celui retourné par la méthode d'analyse des variances. Ces deux valeurs ont

été trouvées expérimentalement. Ceux sont celles qui retournent un score F1 maximal. Si l'on se réfère aux courbes de la Figure 4.6, une diminution du ratio d'agrandissement (autour de la valeur proposée par la méthode des variances), revient à améliorer la précision et diminuer la sensibilité. En d'autres mots, le nombre de Faux Positifs diminue, mais moins de poulets sont détectés. Pour rappel (section 3), il est possible de corriger en post traitement des pertes momentanées de poulets. En revanche, il n'est pas possible de corriger une erreur liée à un échange d'identifiants entre deux poulets. C'est pourquoi il est préférable de se baser sur la méthode de différence de surfaces entre le 15^{eme} et le 85^{eme} quantile afin de déterminer le ratio optimal d'agrandissement de l'image.

4.1.3 Métrique et seuil de détection

Définir une métrique et un seuil de détection permet de déterminer si un poulet est bien détecté ou non. La figure 4.7 représente quatre poulets détectés. Le poulet ayant pour identifiant l'ID 1, est parfaitement détecté. La *bounding box* est parfaitement circonscrite au poulet, ce qui n'est pas le cas pour les trois autres poulets. Le poulet 2 a une *bounding box* bien plus grande que lui, le poulet 3 a une *bounding box* trop petite, et le dernier a sa *bounding box* qui ne l'encadre pas du tout. La question est de savoir à partir de quand peut-on estimer que l'animal à bien été détecté.

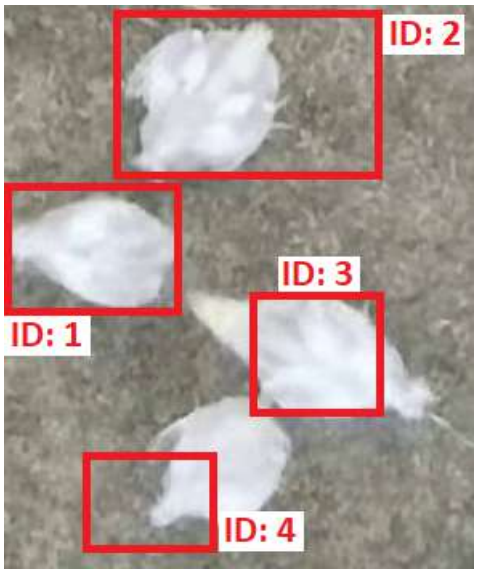

FIGURE 4.7 – Cette figure présente différentes qualités de détection. Le poulet 1 est parfaitement détecté, alors que le poulet 4 voit sa *bounding box* avec un faible taux de recouvrement

La première métrique est appelée *Intersection over Union* (IoU) (voir Figure 4.8). Elle représente le taux de recouvrement en deux *bounding box*. Ces deux *bounding box* sont d'une part la *bounding box* théorique, celle qui est parfaitement circonscrite au poulet, et la *bounding box* issue du réseau de neurones.

Les *bounding box* théoriques ont été tracées à la main afin de pouvoir présenter ces résultats. Ces tracés ont été effectués grâce au logiciel *LabelImg* (rappel Figure 2.30). La figure 4.9 représente en bleu les *bounding box* théoriques tracées à la main, et en rouge celles issues du réseau de neurones. Par convention, une détection est considérée comme bonne si l'IoU est supérieure ou égale à 0.5 et comme très bonnes si celle-ci est supérieure à 0.9. Une IoU inférieure à 0.5 correspond alors à un Faux Positifs. L'IoU traduit à la fois la précision sur la position de l'animal (coordonnées X et Y), et sur la surface de l'animal (hauteur et largeur de la *bounding box*).

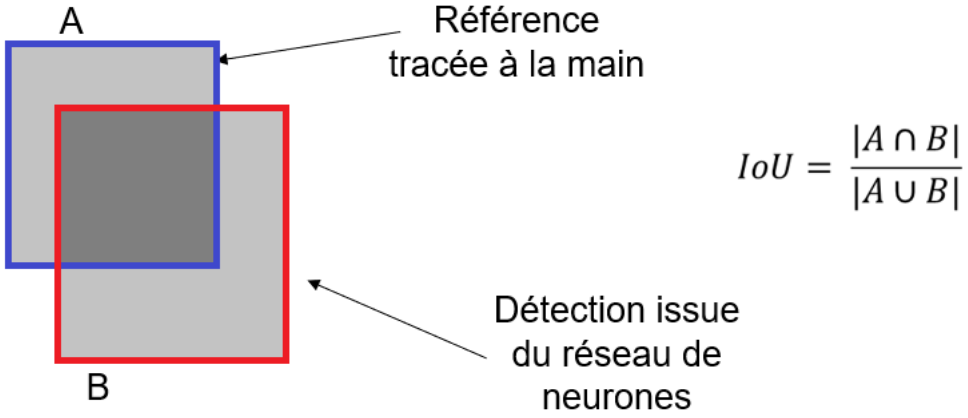

FIGURE 4.8 – Schématisation de la mesure IoU (*Intersection over Union*) entre deux *bounding box*

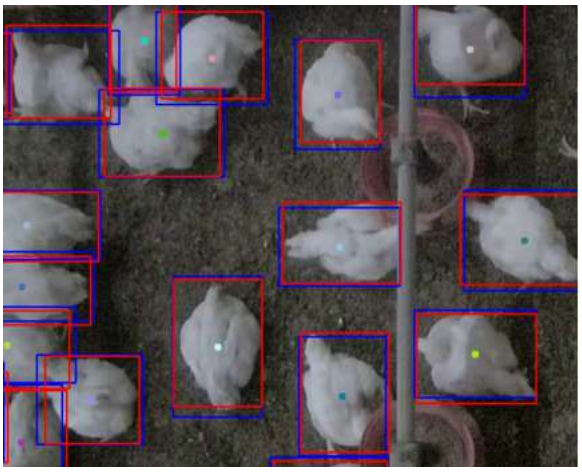

FIGURE 4.9 – Les *bounding box* bleues sont celles de référence tracées à la main. Celles en rouge, sont celles issues du réseau de neurones

La deuxième métrique utilisée se base seulement sur la distance L2 entre la réelle position de l'animal (position théorique) et celle retournée par le réseau de neurones. Ce qui est important pour le suivi de l'animal ce n'est pas tant sa surface, mais bien sa position. La métrique IoU a l'avantage d'être une

	2,8 mètres	3,7 mètres	5 mètres
Sensibilité (IoU)	$67,2\% \pm 0,21\%$	$11,3\% \pm 0,13\%$	$3,1\% \pm 0,05\%$
Sensibilité (Distance)	$77,5\% \pm 0,29\%$	$57,4\% \pm 0,21\%$	$37,5\% \pm 0,14\%$

TABLE 4.4 – Comparaison de la sensibilité de détection entre les deux métriques (IoU et Distance). Ces données correspondent à des poussins de 12 jours. Plus la caméra est haute, plus la surface apparente des poussins est petite. Il apparaît que la condition d'avoir simplement une IoU supérieure à $0,5$ est beaucoup trop drastique.

métrique de référence en détection (une valeur supérieure à $0,5$ représente par convention une bonne détection). A priori, il n'existe pas de valeur de distance (entre une position réelle est celle issue du réseau de neurones) qui garantisse une *bonne* détection. D'un autre côté, si l'on se réfère seulement à l'IoU entre deux *bounding box*, il apparaît que de nombreux poulets qui semblent bien détectés (simple contrôle visuel) possèdent une IoU inférieure à $0,5$. C'est notamment le cas lorsque les poulets ont une surface apparente très petite sur l'image (caméra haute et poulets jeunes). La distance maximale retenue pour une bonne détection est fixée à un tiers de la médiane du grand côté des poulets. C'est la distance maximale qu'il peut y avoir entre deux *bounding box* avec un minimum d'IoU de $0,5$. Le Tableau 4.4 compare la sensibilité de détection obtenue sur des poussins de 12 jours selon trois hauteurs différentes de caméra. Il apparaît par exemple que seuls $3,1\%$ des animaux sont considérés comme bien détectés avec la méthode basée sur l'IoU, là où plus de 37% des animaux voient le modèle de détection retourner une *bounding box* à moins de $33,3\%$ de leur taille.

Le Tableau 4.5 présente la moyenne des IoU des poulets détectés selon trois hauteurs de caméra et trois âges différents. Les images sont prises par une caméra 3M pixels. Les surfaces du champ de la caméra varient de $15m^2$ à $48m^2$ selon la hauteur de la caméra. Il est à noter que la précision de ces résultats dépend à la fois de la précision du modèle de détection, mais également de la précision avec laquelle les *bounding box* de référence ont été tracées. La précision avec laquelle les *bounding box* de référence sont tracées peut être estimée en comparant deux versions d'un même jeu de données de référence. En ce sens, il a été mesuré une moyenne d'IoU de $89,0\% \pm 0,11\%$ entre deux versions de référence pour des poulets de 26 jours et une caméra placée à 2,8 mètres de haut. Ces données sont à comparer à ceux que l'on peut trouver dans la littérature. Dans sa revue *Comparative study on poultry target tracking algorithms based on a deep regression network* (FANG et al., 2020), Cheng Fang compare différentes méthodes de tracking de poulets. On y trouve une valeur d'IoU de $89,1\%$ pour un poulet. Il utilise une caméra de 920×1080 pixels placée à 1,7 mètres de haut pour une surface filmée de $1,65m^2$. Les poulets ont entre 28 et 42 jours et le modèle de détection est un réseau de neurones AlexNet entraîné sur près de 17000 images. Dans notre cas, la situation la plus proche avec les conditions de captation de l'article (FANG et al., 2020)

	J12	J26	J39
2,8 mètres	$74,7\% \pm 0,25\%$	$80,4\% \pm 0,32\%$	$84,7\% \pm 0,24\%$
3,7 mètres	$63,1\% \pm 0,20\%$	$79,6\% \pm 0,15\%$	$82,1\% \pm 0,14\%$
5 mètres	$52,3\% \pm 0,15\%$	$77,6\% \pm 0,12\%$	$77,8\% \pm 0,11\%$

TABLE 4.5 – Intersection over Union entre des données de référence et des données issues du réseau de neurones en fonction de l'âge des poulets et de la hauteur de la caméra

	Mask RCNN	Unet	Mnet	MSAnet	Référence
Sensibilité	77,3%	85,5%	88,2%	90,17%	91,5%

TABLE 4.6 – Comparaison de la sensibilité de détection du Faster RCNN présenté précédemment (section 2.3.2.1), et ceux de l'article (LI et al., 2021)).

reste celle où la caméra est placée 2,8 mètres pour des poulets de 39 jours, avec une résolution d'image 5 fois plus petite (3 Mpixels pour $15m^2$ contre 1 Mpixel pour $1,7m^2$). Les ordres de grandeur entre ces deux résultats (84,7% vs 89,1%) semblent assez similaires.

4.1.4 Sensibilité de la détection

Les tests de détection ont donc été réalisés selon trois âges différents et trois hauteurs de caméras différentes. Pour chaque situation, on compare d'une part la sensibilité suite à une simple détection (Tableau 4.7) avec la sensibilité suite à une triple détection (Tableau 4.8) (voir section 2.3.2.4 pour la triple détection). Dans l'article (LI et al., 2021), Wei Li présente les résultats de la détection de poulets de 4 réseaux neuronaux : Mask-RCNN (HE et al., 2017), Unet (RONNEBERGER, FISCHER et BROX, 2015), Mnet (FU et al., 2018) et MSA-net (LI et al., 2021). Ces réseaux ont été entraînés avec un peu plus de 300 images de 790×930 pixels (exemple Figure 1 de l'article (LI et al., 2021)). Ce sont des images de poulets d'une vingtaine de jours avec un champ de vision d'à peu près $1m^2$. Le cas de figure qui semble se rapprocher le plus de celle de l'article correspond à la captation des poulets de 26 jours avec une caméra à 2,8 mètres de haut. Le Tableau 4.6 représente ces différentes sensibilités. Bien que les conditions ne soient pas les mêmes entre les données utilisées dans l'article et celles dans le modèle Faster RCNN de ce projet, les sensibilités de détection ont le même ordre de grandeur.
Il ressort des résultats des Tableaux 4.7 et 4.8 que la sensibilité de détection est meilleure pour des caméras basses que pour des caméras hautes. De même, celle-ci augmente avec l'âge des poulets. La résolution du poulet dans l'image initiale (nombre de pixels par poulet) semble être fortement corrélée à la sensibilité de la détection. Il semble tout de même que la sensibilité soit plus fortement lié à l'âge des poulets qu'à la hauteur de la caméra, et cela, pour deux raisons :

	J12	J26	J39
2,8 mètres	$84,2\% \pm 0,22\%$ **5181**	$91,2\% \pm 0,23\%$ **12095**	$92,7\% \pm 0,12\%$ **22837**
3,7 mètres	$81,7\% \pm 0,21\%$ **2709**	$90,9\% \pm 0,11\%$ **6205**	$95,1\% \pm 0,08\%$ **8835**
5 mètres	$70,1\% \pm 0,14\%$ **1680**	$90,7\% \pm 0,09\%$ **4018**	$93,6\% \pm 0,06\%$ **5984**

TABLE 4.7 – Sensibilité pour une simple détection. La sensibilité augmente avec l'âge du poulet et inversement avec la hauteur de la caméra. En gras sont présentés les surfaces médianes des poulets en pixel (avant agrandissement de l'image).

	J12	J26	J39
2,8 mètres	$96,9\% \pm 0,09\%$ **5181**	$99,2\% \pm 0,07\%$ **12095**	$100\% \pm -\%$ **22837**
3,7 mètres	$94,2\% \pm 0,15\%$ **2709**	$98,6\% \pm 0,04\%$ **6205**	$99,8\% \pm 0,02\%$ **8835**
5 mètres	$87,9\% \pm 0,12\%$ **1680**	$98,0\% \pm 0,04\%$ **4018**	$99,5\% \pm 0,02\%$ **5984**

TABLE 4.8 – Sensibilité pour une triple détection. La sensibilité augmente avec l'âge du poulet et inversement avec la hauteur de la caméra. En gras sont présentés les surfaces médianes des poulets en pixel (avant agrandissement de l'image)

	J12	J26	J39
2,8 mètres	$23,3\% \pm 0,23\%$ $\mathbf{29,5\% \pm 0,22\%}$	$5,2\% \pm 0,20\%$ $\mathbf{11,6\% \pm 0,27\%}$	$9,7\% \pm 0.80\%$ $\mathbf{18,7\% \pm 0,97\%}$
3,7 mètres	$23,1\% \pm 0,13\%$ $\mathbf{33,9\% \pm 0,13\%}$	$6,4\% \pm 0,09\%$ $\mathbf{17,2\% \pm 0,12\%}$	$6,3\% \pm 0,10\%$ $\mathbf{24,8\% \pm 0,13\%}$
5 mètres	$38,9\% \pm 0,12\%$ $\mathbf{46,9\% \pm 0,10\%}$	$6,6\% \pm 0,07\%$ $\mathbf{17,1\% \pm 0,10\%}$	$5,3\% \pm 0,06\%$ $\mathbf{21,0\% \pm 0,10\%}$

TABLE 4.9 – Tableau du taux de fausses découvertes en fonction de l'âge des poulets et de la hauteur de la caméra. La valeur en gras représente ce même taux mais avant application du seuil de distance minimale ente poulets.

— La base de données qui a permis d'entraîner le réseau de neurones n'est composée que d'images prises avec une caméra placée entre 2 mètres et 2,5 mètres de haut. Or, la sensibilité d'une détection sur une image prise à 5 mètres de haut sur des poulets de 39 jours (93, 6%) est bien meilleure que celle d'une image prise à 2,8 mètres sur des poulets de 12 jours (84, 2).
— Si l'on se réfère à la résolution des poulets (nombre de pixels par poulets), là encore l'âge du poulet a plus d'importance que la résolution. Par exemple, la sensibilité d'une détection sur une image prise à 5 mètres de haut sur des poulets de 39 jours (93, 6%) est meilleure que celle d'une image prise à 3,7 mètres sur des poulets de 26 jours (90, 9%) bien que dans le premier cas leur surface médiane soit 5984 pixels contre 6205 pixels dans le deuxième cas.

Ce qui semble ressortir d'une telle analyse, c'est que les images des jeunes poulets ont moins influencé l'ajustement des poids du modèle Inception lors de l'apprentissage des features que n'ont pu faire des images composées de poulets plus âgés.

4.1.5 Taux de fausses découvertes

Le taux de fausses découvertes représente le nombre de faux positifs en fonction du nombre total de détections. Le Tableau 4.9 représente ces taux de fausses découvertes selon l'âge des poulets et la hauteur de la caméra. Les images de jeunes poulets (12 jours) sont celles qui possèdent le plus de Faux Positifs. Si l'on se réfère à la figure 4.6, on constate qu'il y peu de chances d'avoir un taux inférieur à 25% pour cet âge de poulets. Avec un tel niveau de Faux Positifs il est alors très difficile de faire un suivi de poulets convenable. Il est à noter qu'une grande partie de ces Faux Positifs se trouve sur les mangeoires. Il est donc facile (connaissant la position des mangeoires) de supprimer une partie des Faux Positifs. De plus, il est possible de mesurer la taille médiane des poulets détectés. Cela permet de fixer la distance minimale que l'on autorise entre deux poulets (Chapitre 3) et ainsi supprimer bon nombre de Faux Positifs (un seul Vrai Positif par poulet).

4.2 Suivi

Toutes les données de suivi sont détaillées dans l'annexe B.

4.3 Post-traitement

L'objectif du post-traitement est de corriger un maximum d'erreurs de tracking. Il s'agit alors de traiter les pistes jumelles, supprimer les faux positifs (pistes ne correspondant pas un poulet) et de raccorder les pistes appartenant aux mêmes poulets.

4.3.1 Jumeaux

Pour rappel, la Figure 3.13 schématise la gestion de pistes jumelles. Pour un binôme piste jumelle / piste de référence, l'enregistrement est découpé en trois parties. La première est seulement composée de la piste de référence et se termine à l'apparition de la piste jumelle. La seconde partie correspond à la période commune entre les deux pistes. Enfin, la troisième partie est seulement composée de la piste de référence. Le post-traitement s'occupe de la seconde partie. Le but est de remplacer le morceau de la piste de référence de cette période par la piste jumelle et ainsi former une unique piste correspondant au même animal.

4.3.2 Association

Il est tout à fait normal d'avoir des pistes n'ayant pas la même taille, car le champ de la caméra est un champ ouvert. C'est-à-dire que les animaux peuvent entrer et sortir du champ de la caméra à tout moment. En revanche une piste ne peut pas *mourir* dans n'importe quelle zone de l'image. Une piste doit pouvoir mourir seulement si l'animal qu'elle représente sort du champ de la caméra. Si la piste *meurt* au milieu du champ de la caméra, c'est que l'animal n'a pas pu être suivi correctement. Cela veut dire qu'un point cible dans la région "*proche*" d'où la piste est morte n'a pas pu être affecté à cette piste, et qu'il a sûrement été la source d'une nouvelle piste correspondant au même animal. Ces deux zones sont représentées en vert et en rouge sur la Figure 4.10. La zone verte représente la zone où une piste est autorisée à *mourir*. La zone rouge est la zone dans laquelle une fin de piste doit être raccordée au début d'une nouvelle piste.

Encore une fois, il est question d'associer deux groupes de points. Les débuts de piste d'un côté avec les fins de pistes de l'autre. L'algorithme Hongrois est bien adapté lorsque l'on vise à minimiser un coût total. Pour ce post traitement, il est surtout question d'associer des pistes entre elles indépendamment du coup total que cela peut générer. L'algorithme d'optimisation combinatoire choisi est l'algorithme de Gale-Shapley (GALE et SHAPLEY, 1962). Il

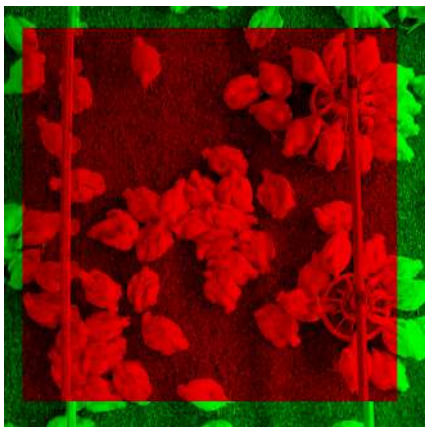

FIGURE 4.10 – L'image représente en vert la zone dans laquelle une piste peut mourir et en rouge la zone dans laquelle une fin de piste doit être raccordée au début d'une nouvelle piste

se base sur les distances en temps et en espace entre les fins et les débuts de pistes candidats afin de créer un classement de préférence d'affectation pour chaque piste. image gaussienne affectations

4.4 Indicateurs

À partir de l'enregistrement des positions et des dimensions des poulets au cours du temps, il est alors possible créer tout un ensemble d'indicateurs de comportement animal. Ces indicateurs dépendent des données issues du tracking ainsi que des données basées sur l'environnement des poulets (zones mangeoires, abreuvoirs et zones perchoirs). Le fichier crée en sortie du tracking comprend donc des temps, des identifiants, des positions, des bounding box et des valeurs de pixels pour chaque point enregistré. Toutes ces données permettent alors de créer des indicateurs représentatifs du comportement des animaux. Les indicateurs créés sont les suivants :

— **Le temps passé aux mangeoires.** Une zone circulaire autour de chaque mangeoire a été estimée expérimentalement. Un animal dont le centre est enregistré à l'intérieur de cette zone est considéré comme ayant accès à la mangeoire. On connaît donc pour chaque poulet, le temps et la fréquence à laquelle celui-ci se rend à une mangeoire.
— **Le temps passé aux abreuvoirs.** De la même façon que pour les mangeoires, des zones ont été estimées autour des abreuvoirs, à l'intérieur desquelles un poulet est considéré comme ayant accès à l'abreuvoir.
— **Les distances parcourues.** Connaissant la position de chaque animal au cours du temps, il est alors possible de mesurer les distances parcourues pour chaque animal. Celles-ci sont reportées en centimètres.
— **Les vitesses atteintes.** Tout comme pour les distances mesurées précédemment, il est possible de mesurer les vitesses (nombre de pixels par image). Celles-ci sont traduites en centimètres par seconde.

— **Les surfaces.** La surface de chaque poulet se traduit à travers la surface de leur bounding box. Une analyse a été portée sur la relation qu'il existe entre la surface de ces bounding box et le poids des animaux (voire section Résultats).
— **Les espaces disponibles.** L'espace disponible de chaque animal est représenté par la surface de sa cellule de Voronoï (cellules dont les arrêtes correspondent aux médianes entre chaque poulet).
— **Espace visité.** Cet indicateur est un complément de celui sur les distances parcourues. Cet indicateur traduit le côté explorateur du poulet, à savoir si ses déplacements sont concentrés au même endroit de l'image ou pas. Pour cela l'image est découpée en 256 zones (valeur déterminée expérimentalement). Puis l'on relève le nombre de cases visitées.
— **L'activité.** Un poulet est considéré actif lorsque celui-ci se déplace, et inactif lorsque sa position de bouge pas. Pour cela un lissage est effectué sur ses positions, et un seuillage de déplacement a été estimé. Pour tout déplacement supérieur à ce seuil, l'animal est considéré comme en déplacement. Ce lissage et ce seuil permettent de s'affranchir des variations de mouvement dues à l'imprécision de la détection.

Avec de telles données, il est possible de mesurer des indicateurs individuels (Figure 4.11) ou des indicateurs globaux (Figures 4.12 et 4.13).

4.4.1 Surfaces

La connaissance du poids des poulets est facteur très important dans un élevage. La distribution de ces poids et leur évolution au cours du temps en dit beaucoup sur l'état de santé du lot. Une première expérimentation a été portée en conditions expérimentales afin d'établir une possible correspondance entre la surface des poulets (que l'on peut mesurer) et leur poids (que l'on cherche à connaître). La Figure 4.14 présente les résultats de cette expérimentation. Tous les trois jours une dizaine d'animaux ont été pesés et placés sous la caméra afin de mettre en correspondance leur poids avec leur surface apparente sous le champ de la caméra. Il en a été conclu que l'analyse de la surface des animaux était un bon indicateur qui permet de traduire le poids des animaux. Il est alors possible d'analyser la distribution de ces surfaces au cours du temps de les mettre en correspondance avec des courbes de référence qui doivent être calibrées de façon expérimentale. Les Figures 4.12 et 4.13 représentent ces surfaces à deux semaines d'intervalles. À noter que ce type d'indicateur ne nécessite pas de suivi des animaux, mais seulement une détection des poulets.

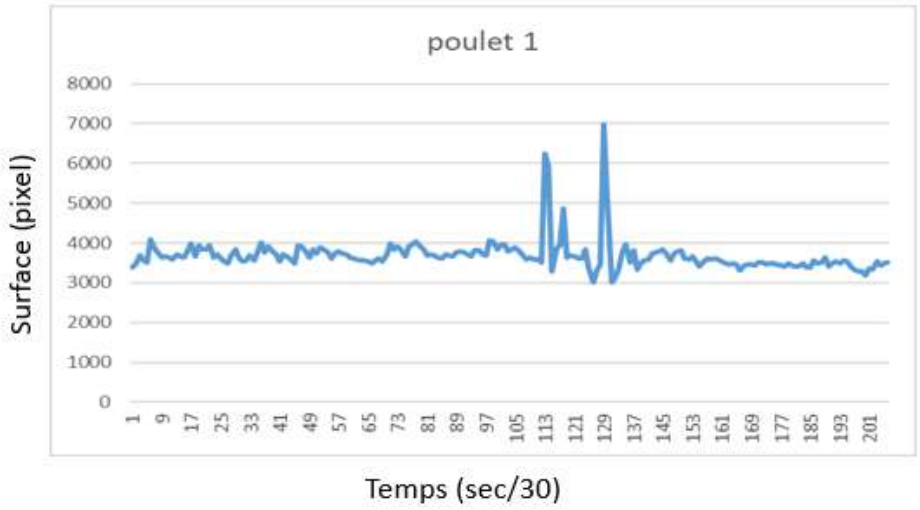

FIGURE 4.11 – Évolution de la surface d'un poulet au cours du temps. Les deux pics correspondent aux battements d'ailes du poulet.

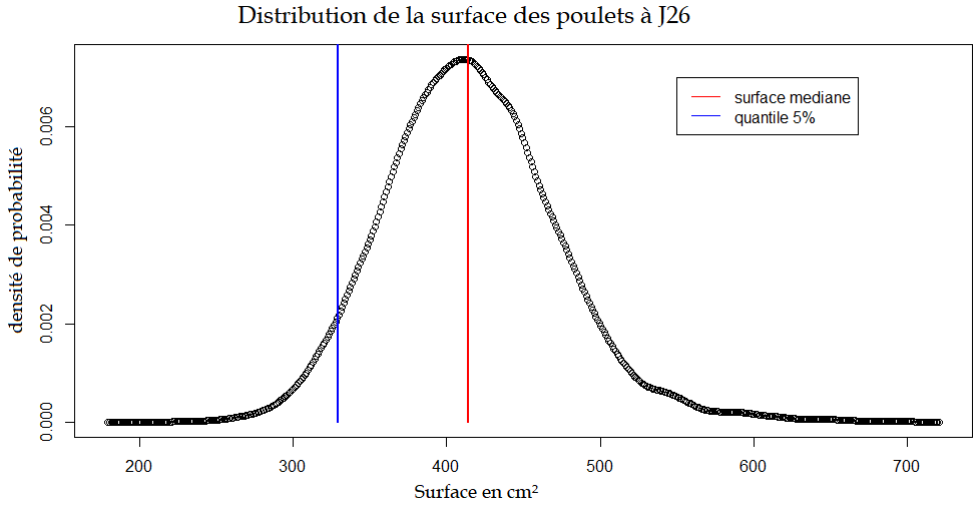

FIGURE 4.12 – Distribution de la surface des poulets à 26 jours.

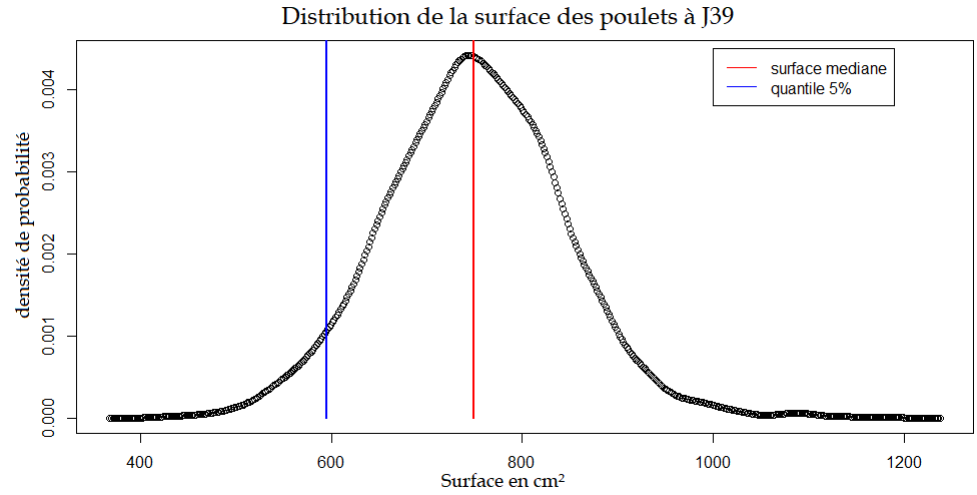

FIGURE 4.13 – Distribution de la surface des poulets à 39 jours.

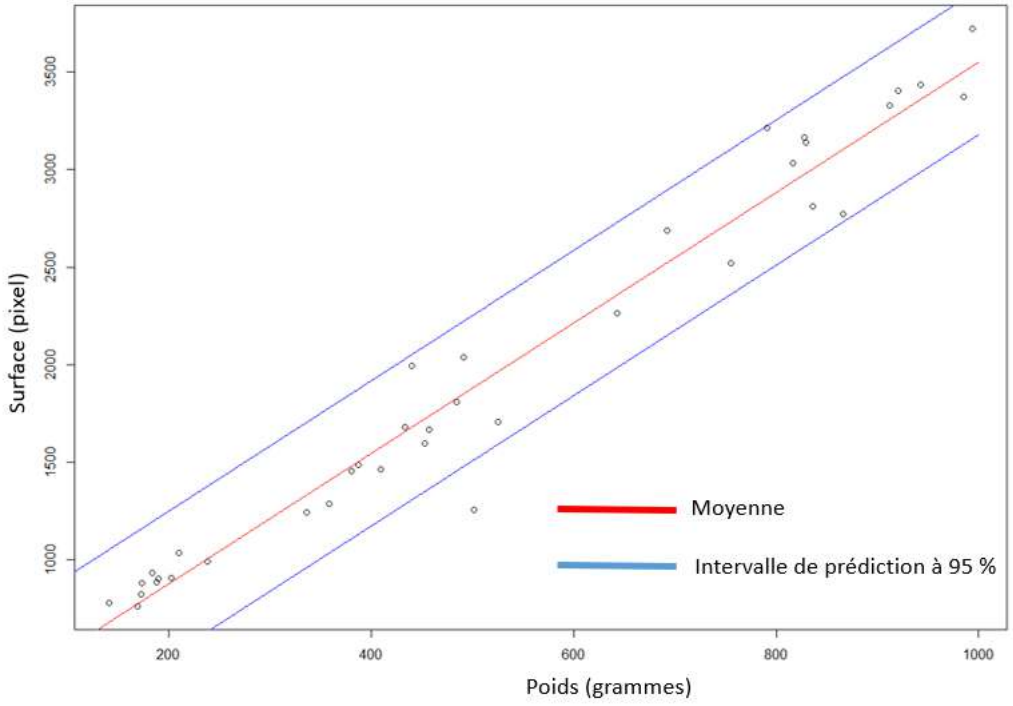

FIGURE 4.14 – Analyse de la surface des poulets. La surface de la *bounding box* est ici mise en correspondance avec le poids des animaux.

4.4.2 Autres indicateurs

Afin d'être parlante, une courbe caractérisant un indicateur doit être mise en correspondance avec une référence. Cette référence, qui vient généralement avec des limites d'acceptabilité des données (intervalle de confiance) permet de dire si les données des indicateurs sont acceptables. Puisque la définition de ces références ne rentre pas dans le cadre de ce projet, il n'y a donc pas

de résultats à proprement parlé des différents indicateurs. Voici une liste non exhaustive des différents indicateurs qui peuvent être créés :

— Distribution des surfaces (Figures 4.12 et 4.13)
— Distribution des temps passés à la mangeoire, à l'abreuvoir ou aux zones perchoirs (Figure 4.15)
— Distribution du nombre d'animaux présents à la mangeoire, à l'abreuvoir ou aux zones perchoirs.
— Distribution des distances parcourues
— Distribution du nombre d'animaux en déplacement (Figure 4.16)

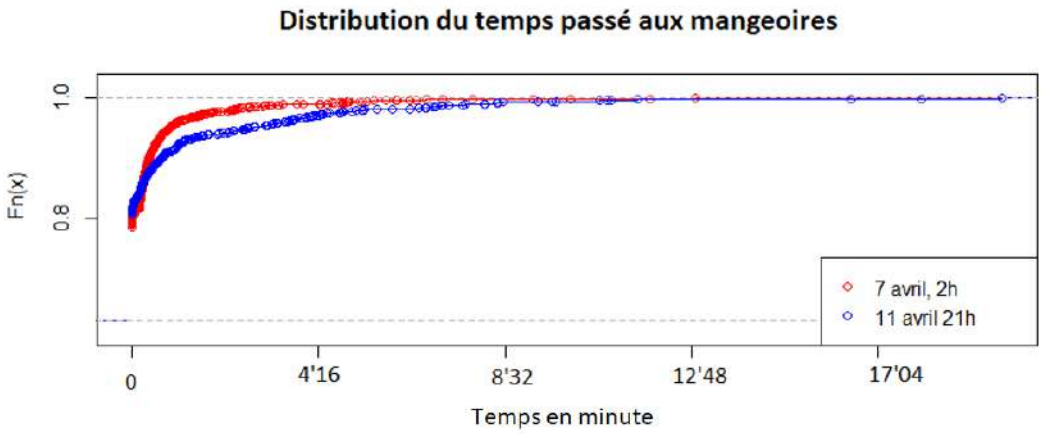

FIGURE 4.15 – Comparaison des distributions des temps passés à la mangeoire à 4 jours d'intervalle et différentes heures de la journée sur des vidéos de 2h

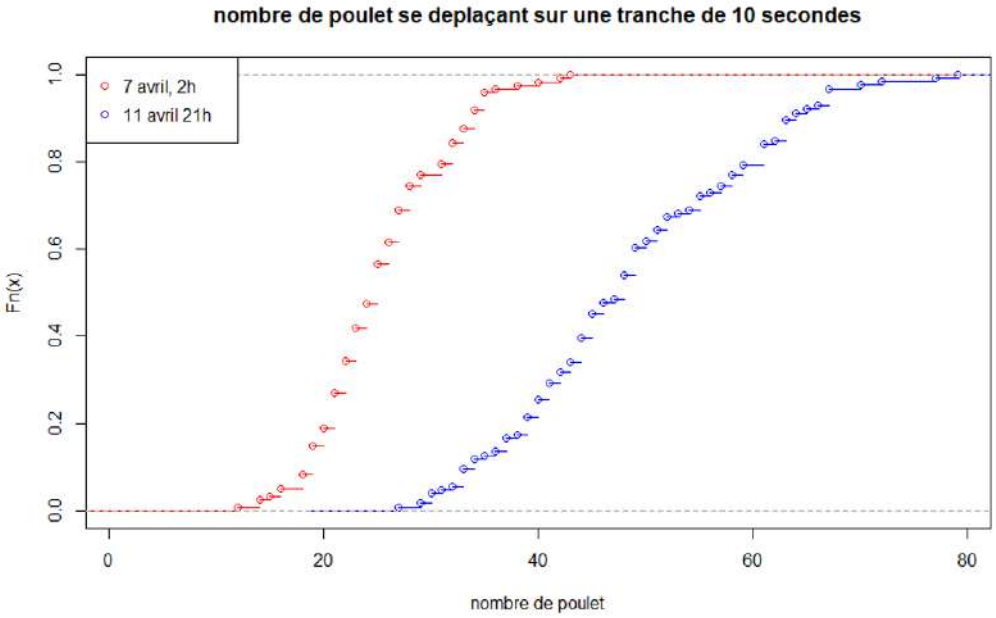

FIGURE 4.16 – Comparaison des distributions du nombre d'animaux en déplacement à 4 jours et différentes heures de la journée d'intervalle sur des vidéos de 2h

Enfin, il est possible de créer quelques indicateurs supplémentaires, indépendamment des données de tracking, mais directement des images enregistrées. Par exemple, en retenant sur l'ensemble de la vidéo les valeurs minimales des pixels à chaque endroit de l'image, il est possible de créer une carte d'intensité dont les valeurs de pixels les plus élevées traduisent une absence de mouvement sur l'ensemble de la vidéo (Figure 4.17). Cette technique permet notamment de détecter les poulets morts sous le champ de la caméra. Il est également possible (moyennant des images de référence) d'évaluer l'état de la litière. La Figure 4.18 représente la comparaison qui peut être faite entre plusieurs états de litières par rapport à des imagettes de référence (section 2.1.2.2).

FIGURE 4.17 – Détection des poulets immobiles. À gauche l'image initiale, à droite l'image d'intensité après 5 minutes.

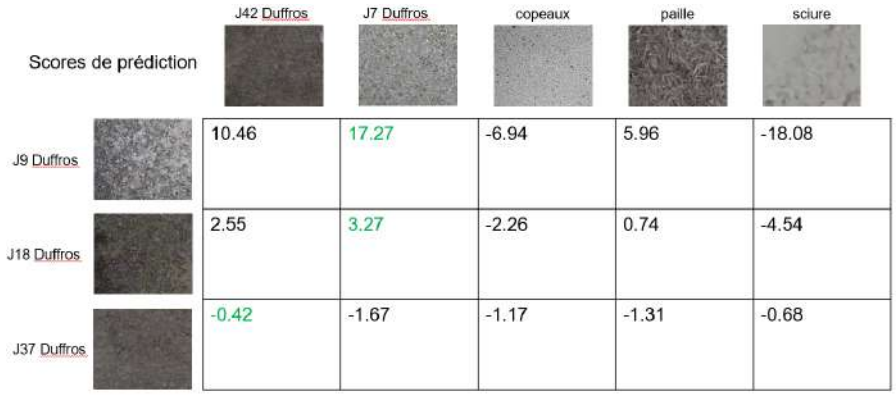

FIGURE 4.18 – Comparaison de l'état de 5 litières avec 3 litières de référence.

Chapitre 5

Conclusion

Le but de cette thèse a été de développer un outil de tracking de poulets de chair en élevage commercial. Pour ce faire, un ensemble de tests ont été réalisés, tout d'abord dans un environnement contrôlé (expérimental) puis en élevage commercial. Il a été établi la meilleure façon de capter les images. Un modèle de réseaux de neurones a ensuite été entraîné à partir d'une base de données d'images issues des différentes captations réalisées. À la suite du développement de cet outil de détection de poulets de chair, un algorithme de suivi de poulets a été développé afin de générer un ensemble de données sur la position des poulets et leur identification sous le champ de la caméra au cours du temps. Enfin, il a été développé quelques algorithmes qui permettent de traduire ces données brutes de tracking sous forme de distribution d'indicateurs qui permettent notamment une meilleure lecture du comportement global des animaux.

État de l'outil développé Bien que cet outil ne soit pas capable d'effectuer un parfait suivi des animaux sans aucune erreur, il a été relevé les conditions optimales pour lesquelles le tracking est actuellement le plus efficace. Les conditions expérimentales permettent de s'affranchir de certains problèmes comme les variations d'illumination et les entrées/sorties des animaux du champ de la caméra. Puisque l'analyse se porte sur un nombre beaucoup plus faible d'animaux, la caméra est placée plus basse qu'en condition commerciale et la résolution des images se trouve alors bien plus élevée. Dans la mesure où le réseau de neurones a principalement été entraîné sur les images des captations faites en milieu expérimental, une bonne détection nécessite alors une bonne résolution des images. Afin de garder la même résolution avec une caméra plus haute, il faudrait alors augmenter la définition de l'image. À noter qu'une augmentation de la définition d'une image traduit un agrandissement de la taille de cette image et ainsi du nombre de pixels à traiter. La définition d'une image agit directement sur le temps de traitement de cette image qui est un paramètre important à prendre en compte si l'on tient par la suite à travailler en temps réel. On a donc outil qui est capable de générer plusieurs indicateurs de toutes sortes. Certains de ces indicateurs proviennent directement de ceux que l'on retrouve dans la méthode EBENE® comme le nombre d'animaux aux mangeoires, d'autres comme la mesure de l'espace disponible entre poulets sont de nouveaux indicateurs qui viennent enrichir ces indicateurs qui permettent d'estimer le bien-être des animaux. Le passage

de ces indicateurs sous forme de distribution est également une évolution importante dans l'analyse des données et qui est un vrai plus par rapport aux données de la méthode EBENE®. Travailler avec une distribution permet de se focaliser sur les queues de cette distribution. Les queues de la distribution traduisent les animaux qui s'écartent de la moyenne. Plus ces extrémités de distribution sont courtes, plus l'ensemble des animaux traduisent le même profil. Ce que recherche généralement l'éleveur est une uniformité des comportements des animaux au sein de l'élevage. Bien qu'il existe actuellement des outils qui permettent d'estimer par exemple la surface des poulets ou de suivre des poulets en conditions expérimentales, notre outil semble bien être le premier à pouvoir réellement apporter une aide aux éleveurs.

Utilisation des indicateurs et perspectives On a donc un outil qui permet de mesurer toutes sortes d'indicateurs. À noter que ce projet ne s'est pas penché sur la définition d'un bon ou d'un mauvais indicateur. En d'autres mots, ce projet a permis de tracer des courbes de distribution à partir de vidéos capter en élevage commercial. Mais à l'issu de ce projet, ces courbes de distributions n'ont pas encore été calibrée. On ne sait pas à partir de quand est-ce qu'une courbe traduit un élevage en état de bien être ou un élevage dans lequel il y a une maladie ou problème technique qui apparaît. Cette problématique doit être étudiée lors du projet suivant appelé Ebroiler Track 2.0. Il sera alors question de fixer des limites acceptables de ces distributions ainsi que les incertitudes de mesure. Pour cela des élevages seront très régulièrement suivi par des vétérinaires et évaluer avec la méthode EBENE® déjà existante. Dans le même temps L'élevage sera filmé afin de pouvoir faire le comparatif avec les résultats de la méthode EBENE®. Cette technique d'évaluation du comportement des animaux peut également être recoupé avec d'autres techniques afin d'enrichir les données utilisées dans la méthode EBENE®. La méthode par analyse d'image n'est pas la seule développée dans le cadre du projet Ebroiler-Track. Une méthode de captation sonore en élevage a également été développée afin de détecter certaines maladies qui se traduisent par un changement du piaulement des animaux.

Limites de l'outil et améliorations envisagées Un enrichissement de la base de données permettant d'entraîner le réseau de neurones fait partie des priorités afin d'améliorer les performances de cet outil de tracking. Cela permettrait notamment de détecter avec plus de justesse les poussins ainsi que d'améliorer les captations faites par des caméras prises à 5 mètres de haut.
Le temps de traitement est également un critère très important dans l'industrie. Actuellement il n'est pas possible de traiter les données en temps réel, ce qui pourrait être une vraie plus-value pour envoyer un message à l'éleveur en cas de problème rencontré. Toutes les captations dans le cadre de ce projet ont été faites sur des vidéos à 10 images par seconde. Il semblerait que réduire la fréquence de captation lors des périodes de faible activité des animaux puisse être une bonne opération.

De plus, améliorer la détection la base de données pour des détections faites sur des images prises à 5 mètres de haut permettra d'éviter d'appliquer un zoom sur les images. Éviter d'appliquer un zoom veut dire travailler avec des images plus petites et ainsi réduire le temps de calcul. S'il est possible d'améliorer efficacement la base de données, il sera peut-être possible d'éviter d'appliquer une triple détection et ainsi gagner encore du temps de calcul.
Ce projet a eu pour but de développer un outil de suivi des animaux, le critère temps d'exécution n'était que secondaire. Les codes ayant été développés en Python, il peut être possible d'accélérer les temps de traitement en traduisant les codes en langages compilés.

Un deuxième point d'amélioration à étudier correspond aux zones stratégiques à étudier de l'élevage. Les caméras utilisées lors de ce projet sont des caméras de vidéo surveillance d'une largeur de champ de 90°. Une caméra d'une telle largeur de champ placée au plus haut de l'élevage (5 mètres de haut) ne couvre pas toute la largeur du bâtiment. Deux solutions peuvent être proposées : placer une deuxième caméra dans la largeur ou bien augmenter la largeur de champ de la caméra. Puisque le toit est en pente, placer deux caméras dans le sens de la largeur veut dire qu'elles sont alors placées à des hauteurs plus basses ce qui implique que leur surface filmée au sol est alors réduit. Dans ce cas, il n'est toujours pas possible de filmer toute la largeur de l'élevage. Donc soit on place près de 6 caméras pour filmer toute la largeur d'un élevage multiplié par 30 caméras dans le sens de la longueur afin de couvrir entièrement l'élevage, soit on augmente le champ de vision de la caméra. Augmenter le champ de vision de la caméra entraîne une augmentation des déformations optiques encore plus importantes qui peuvent être très compliquées à corriger.
La conclusion d'une telle analyse traduit le fait qu'il est très compliqué de couvrir l'ensemble de l'élevage, sans parler des problèmes liés au recouvrement entre champs de vision des différentes caméras, de leur synchronisation, et du temps de calcul nécessaire à traiter l'ensemble d'un élevage. La solution actuelle est de se limiter à une région de l'élevage qui soit le plus représentatif possible de l'ensemble de l'élevage. Il faut donc pouvoir localiser une zone de l'élevage dans laquelle l'ensemble des animaux puisse exprimer leur comportement naturel. Dans l'idéal, il faut donc : une zone abreuvoir, une zone perchoir, une zone mangeoire, un mûr. Cette zone d'intérêt doit également être un lieu dans lequel les incidents techniques ou les problèmes matériel ont la plus grande probabilité d'apparition. Si l'on veut pouvoir détecter un mal fonctionnement dans l'élevage, il est préférable de le détecter avant qu'il y ait une répercussion à l'ensemble du bâtiment. Une telle zone peut par exemple se trouver près d'une source d'aération ou à la fin d'une ligne de mangeoire ou d'abreuvoir. La recherche d'une telle zone d'intérêt fait partie des champs de recherche du prochain prochain projet Ebroiler Track 2.0.

Récompense et publication Ce projet Ebroiler-Track a été récompensé le 2 mars 2022 au stand du ministère de l'Agriculture et de l'alimentation lors du salon international de l'agriculture à Paris du prix de la 4e édition d'Ita'Innov dans la catégorie Sécurité sanitaire - Santé des plantes et des animaux comme outil d'alerte du bien-être et de la santé des poulets d'élevage de chair. Ce prix récompense les instituts techniques pour leurs travaux dans les domaines de l'innovation agricole, de la production à la transformation alimentaire et non alimentaire.

La méthode de suivi a fait l'objet d'une soumission d'article (en cours) dans le journal *Computers and Electronics in Agriculture* qui couvre les avancées dans le développement et l'application de matériel informatique, de logiciels, d'instrumentation électronique et de systèmes de contrôle pour résoudre les problèmes en agriculture, agronomie, horticulture et élevage. Cet article a été soumis sous le titre de *Optimal Estimation of Broiler Movement for Commercial Tracking*.

Annexe A

Code associé au projet

A.1 Fichier de configuration du modèle Faster R-CNN

```
model {
  faster_rcnn {
    num_classes: 1
    image_resizer {
    keep_aspect_ratio_resizer {
        min_dimension: 800
        max_dimension: 1000
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        height: 128
        width: 128
        scales: [0.125, 0.25, 0.5, 1.0]
        aspect_ratios: [1.0, 1.25, 0.75]
        height_stride: 8
        width_stride: 8
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
```

```
      truncated_normal_initializer {
        stddev: 0.01
      }
    }
  }
  first_stage_nms_score_threshold: 0.0
  first_stage_nms_iou_threshold: 0.6
  first_stage_max_proposals: 200
  first_stage_localization_loss_weight: 2.0
  first_stage_objectness_loss_weight: 1.0
  initial_crop_size: 12
  maxpool_kernel_size: 2
  maxpool_stride: 2
  second_stage_box_predictor {
    mask_rcnn_box_predictor {
      use_dropout: false
      dropout_keep_probability: 1.0
      fc_hyperparams {
        op: FC
        regularizer {
          l2_regularizer {
            weight: 0.0
          }
        }
        initializer {
          variance_scaling_initializer {
            factor: 1.0
            uniform: true
            mode: FAN_AVG
          }
        }
      }
    }
  }
  second_stage_post_processing {
    batch_non_max_suppression {
      score_threshold: 0.001
      iou_threshold: 0.6
      max_detections_per_class: 200
      max_total_detections: 200
    }
    score_converter: SOFTMAX
  }
  second_stage_localization_loss_weight: 2.0
  second_stage_classification_loss_weight: 1.0
}
```

```
}

train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0005
          schedule {
            step: 200000
            learning_rate: 0.0001
          }
          schedule {
            step: 400000
            learning_rate: 0.00001
          }
          schedule {
            step: 600000
            learning_rate: 0.000001
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: PATH to model.ckpt
  from_detection_checkpoint: true
  #freeze_variables: ".*FeatureExtractor.*"
  load_all_detection_checkpoint_vars: true
  #num_steps: 500000
  data_augmentation_options {
    random_rotation90 {
    }
    random_horizontal_flip {
    }
    random_vertical_flip {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
    input_path: PATH to train.record
```

```
    }
    label_map_path: PATH labelmap.pbtxt"
  }

  eval_config: {
    metrics_set: "coco_detection_metrics"
    #use_moving_averages: false
    #min_score_threshold: 0.1
    num_examples: 200
  }

  eval_input_reader: {
    tf_record_input_reader {
      input_path: PATH to test.record
    }
    label_map_path: PATH to labelmap.pbtxt
    shuffle: false
    num_readers: 1
  }
```

Annexe B

Papier

B.1 Abstract

Nowadays, video tracking has taken a considerable part in monitoring systems. It allows identifying and follow every object in the camera field over time. While most of these algorithms are rather well suited to regular movements (following cars, pedestrians), they are often limited in more complex situations (high variations in speed, low detection rate, frequent shape variation). This paper proposes three methods adapted to broilers tracking in commercial environment. Past movements analysis of known broilers enable to estimate their motions and therefore to predict their new position. New unidentified broilers positions are then compared to these predicted positions. Distances between these two sets of positions are then used in the Hungarian Algorithm to assign an ID to new detected broilers regarding their past positions. Our methods differentiate by the way they predict the future positions. Contrary to most methods, they do not seek perfect regularity of movements and can deal with low rate detection. The proposed methods showed better performances than existing one.

B.2 Introduction

One of the most widely used techniques in the field of crowd surveillance remains video analysis. It is a non-invasive and inexpensive method for monitoring objects. Video tracking makes it possible to detect and follow each object in video to document their activities. There are today many tracking methods. These depend on the nature of the tracked objects, the time constraints, and the resources available for the calculations. We are interested in the tracking of several hundred of broilers. Also, the natural framework is the so-called Multiple Object Tracking (MOT) which can deal with many objects compared to single object tracking. Video tracking consists in putting the same ID on the same object on all subsequent images. In his review, (**luo2021multiple**) distinguished the tracking methods according to the way the initialization, the processing steps and the pairing step are performed. There are mainly two ways to perform the initialization step : either a detection of all objects is performed at each image (tracking by detection) or at each image one keeps the objects already present in the previous images. This last way to proceed

characterizes the detection-free tracking methods. As it does not detect new objects entering the camera field, it is not suited to broilers tracking that are not enclosed in the camera field. In the tracking by detection approach (HAN et al., 2004), each object is limited by a bounding box which returns the position and size of the detected object. Detection requires searching objects in the whole image and it is generally applied to well-defined classes of objects. These methods guarantee a pretty good localization of detected objects. The effectiveness of MOT models largely depends on the detection model that precedes them. Nowadays, convolutional neural network detection methods achieve excellent detection rates, and some of them can be used for real-time analyses (ZHAO et al., 2019), (DHILLON et VERMA, 2020). They are preferred to simple image processing methods like those summarized in (BALAJI et KARTHIKEYAN, 2017) and (KOTHIYA et MISTREE, 2015). These methods are highly dependent on environmental conditions like illumination, background contrast, noise in the Image. Two processing modes can be considered ; namely, the online (HUA, ALAHARI et SCHMID, 2015) (WANG et al., 2019) and the offline modes(LUITEN, ZULFIKAR et LEIBE, 2020). When using the online mode, the images are analyzed sequentially, one by one according to their recording time. In order to identify object on an image, the tracking algorithm can only refer to past detected objects. The second mode is offline tracking. Even though it is still possible to identify objects sequentially, in offline mode, detection is realized over the whole image beforehand. It means that for identification, tracking algorithm has access to past identified objects and future detected objects. Where online mode is well suited to real time process, offline mode has the advantage of having more detection data at its disposal, enabling thus to make better allocation decisions. In practice offline mode does not deal with all data but only with a batch of frames. Despite its potential better performances, this mode leads to too long calculations, running out of memory, and a delay in the results output ((**luo2021multiple**), (WEI et al., 2007)).

The two pairing modes are : Deterministic tracking and Stochastic tracking (SETHI et JAIN, 1987) (VEENMAN, REINDERS et BACKER, 2001). Deterministic tracking seeks for the best paring between new detected objects and previous identified objects. To doing so, a metric is chosen in order to favor specific displacements like constant motion. Hungarian algorithm (**kuhn1955hungarian**) is still one of the main cited and used method for such an assignment problem. Greedy algorithms (SALARI et SETHI, 1990) (RANGARAJAN et SHAH, 1991) (VEENMAN, REINDERS et BACKER, 2001) are the mains substitutes to the Hungarian algorithm. Stochastic tracking assumes noises in data coming from the detection model. Therefore, it introduces uncertainty in the measures. These methods are often based on Kalman filter algorithm (LI et al., 2010) (WENG, KUO et TU, 2006). The main stochastic tracking algorithms are Network flow (ZHANG, LI et NEVATIA, 2008), Shortest path (**berclaz2011multiple**) and conditional random field (**milan2015joint**), they are all suited for offline mode tracking. The two widely used methods for multi objects tracking are Joint Probability Data Association Filtering (**hamid2016joint**) and Multiple

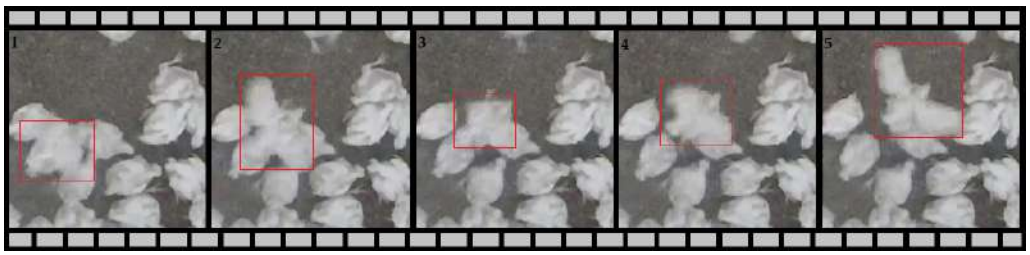

FIGURE B.1 – Silhouette broiler evolution, 20 frames/second

Hypothesis Tracking (BLACKMAN, 2004). The Multi Target Tracking (MTT) try all possible associations between targets and tracked objects over time. This method leads to heavy computation. The Joint Probability Data Association Filtering (JPDAF) is less computationally expensive as it uses a filtering process based on the distance between target and tracked objects to reduce the number of potential association. The main drawback is that JPADF works only for a fix number of tracked objects and struggles to dissociate too closed objects. Main stochastic methods are not suited to unfixed and high number of objects to track.
While most of tracking algorithms are rather well suited to regular movements (following cars, pedestrians), they are often limited in more complex situations with high variations in speed, low detection rate and frequent shape variation. This article aims at developing a tracking method for broilers. This is why, we opt for the use of a detection based method that provides a solution to incoming and out-coming broilers in the camera field. In order to reduce the complexity of calculation due to the high amount of broilers in the camera field and therefore the high number of possible data association, we opt for an sequential processing (that has the advantage to work for both online and offline modes) and a deterministic pairing mode. This tracking method seeks to be independent to animals activity, to their spacing, to their weak movement regularity, to their occlusions etc. More precisely, we present three different ways to predict broilers positions.
The next section summarizes the works published on multi object tracking. The main algorithms used as benchmark to compare to our methods and the description of tree methods we suggest for broilers tracking are presented. Finally, the performances of the tracking methods suggested in this article are compared to the benchmark methods.

B.3 Related work

As reported by (YILMAZ, JAVED et SHAH, 2006), tracking algorithms can separated into three groups according to the information used to identify object in each image. The main idea behind all these algorithms is to select specific object information that remains similar into successive frames for each object, or at least that is easily predictable among successive frames.

kernel tracking algorithms use information like distribution of colors or object's texture included in the bounding box from previous detection. They are usually represented as histograms. The idea is to find object having the "closest" histogram (COMANICIU, RAMESH et MEER, 2003) in successive frames. An example of distance measurement between two histograms is given by (PELE et WERMAN, 2010). For the texture analysis, Tuceryan (TUCERYAN et JAIN, 1993) summarizes the different kinds of features we can find in the literature. At last, template tracking methods are also part of kernel tracking family. It's a brute method looking for correlations between small images delimited by bounding boxes (SCHWEITZER, BELL et WU, 2002). These kinds of methods are really sensitive to illumination.

Silhouette tracking : While kernel tracking deals only with the information included in a fix shape (bounding box), silhouette tracking treats with object's shapes and contours. Shapes tracking seeks to match objects having closed silhouettes in two consecutive frames based on edge map image. If tracked objects are nonrigid, the silhouette has to be updated on each frame (HUTTENLOCHER, NOH et RUCKLIDGE, 1993) (BERTALMIO, SAPIRO et RANDALL, 2000). The contour can be seen as a surface that needs to be matched to previous surface features. In this way, (HUTTENLOCHER, NOH et RUCKLIDGE, 1993) uses the Hausdorff distance in order to evaluate the distance between two surfaces. On the other hand, the contours can be represented as a closed outline. If so, one of the most suitable methods to deal with such feature is the Freeman chain code (VADDI et al., 2011).

Point Tracking : These methods are usually used for small objects or objects with few details. There are deterministic methods (VEENMAN, REINDERS et BACKER, 2001) and probabilistic methods (BAR-SHALOM, DAUM et HUANG, 2009). Deterministic methods define a cost according to the distance between two points (in two different frames). It could be a spatial distance, a difference between two speeds, or anything else that can be computed thanks to recorded positions over time. Then a combinatorial optimization algorithm is used to combine pairs of points in order to minimize the cost sum. The Hungarian algorithm (**kuhn1955hungarian**) is the mainly used algorithm to solve such problem. However, greedy algorithms like those from (SETHI et JAIN, 1987) and (RANGARAJAN et SHAH, 1991) are also widely used.

In our application, we have few choices on the features to use. Our situation is to track several hundred chickens in a farm. Chickens are filmed from above by a camera set upright. All chickens in the video are the same age and have similar colors (Figure B.2). However, their shape that are quite similar when they are resting, can change very quickly as soon as they are moving (Figure B.1). We can then reasonably exclude the features based only on the visual part of the object, as they are weakly discriminative between chickens or are difficult to predict. As chickens have quite the same appearance, we choose the object's position as the main feature representing the chicken.

Tracking of objects summarized by their positions is really sensitive to miss detections. It can end up locally with no points (miss detection) or with one point for many objects (occlusion). One way to correct these problems is to

FIGURE B.2 – Example of input image

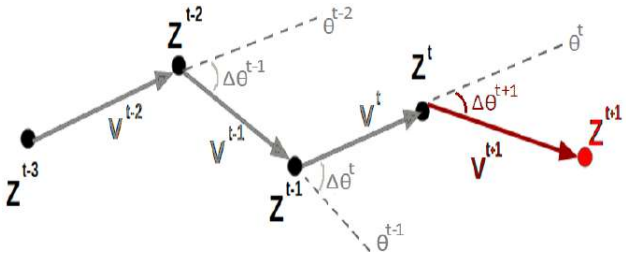

FIGURE B.3 – nomenclature

assess object movement and therefore to predict its future positions as accurately as possible. Most of the methods deal with good detection, no entrance objects and straight movements. While an assumption like a straight movement (RANGARAJAN et SHAH, 1991) suit quite well for objects like cars, it becomes less usable for chickens due to their jerky movement.

B.3.1 Definitions

Let us first suppose that the system is closed, that is, all the objects remain in the field of the camera and no object comes out. In addition, we assume a perfect detection. In other words, all objects present are detected. If n_t denotes the number of detected broilers at time t, these assumptions ensure that $n_0 = \ldots = n_{t_{max}} = N$. It should be noted that at time $t = 0$, there are only detections and arbitrarily ID affectations. The first ID association arises at time $t = 1$. Let's denote by Z_i^t the i object's position in $\mathcal{R}^2$ at time t. In our simplified situation (closed system), performing tracking amounts to build a sequence of t_{max} permutations $(\sigma^t)_{t \leq t_{max}}$ where for all $t \in 0, \ldots, t_{max}$, $\sigma^t \in S_N$, so that for all $i \leq n$ two successive positions of the sequence are "close". The permutation σ^t is the permutation to be applied to the objects of the image $t-1$ to obtain the number of the object on the image t. In the following, we note σ^{-t} the reciprocal permutation of σ^t such that $\sigma^{-t} \circ \sigma^t = \sigma^t \circ \sigma^{-t} = Id$, and $\sigma^{t,1}(i)$ is the contracted form of $\sigma^t \circ \ldots \circ \sigma^1(i)$

$$Z_i^0 \to Z_{\sigma^1(i)}^1 \to \ldots \to Z_{\sigma^{t,1}(i)}^t$$

$$Z^0_{\sigma^{-1,-t}(j)} \leftarrow \ldots \leftarrow Z^{t-1}_{\sigma^{-t}(j)} \leftarrow Z^t_j$$

In the above equation, $i = \sigma^{-1,-t}(j)$ and $j = \sigma^{t,1}(i)$.
The velocity vector $\mathbf{V}^t_{i,k} = Z^t_k - Z^{t-1}_i$allows to go from the point Z^{t-1}_i to Z^t_k is noted . When these points belong to the path of the same broiler like $Z^{t-1}_{\sigma^{-t}(i)}$ and Z^t_i this notation is simplified to $\mathbf{V}^t_i$ instead of $\mathbf{V}^t_{\sigma^{-t}(i),i}$. Its magnitude corresponds to the speed of the object. The angle between two successive velocity vectors $\mathbf{V}^{t-1}_{\sigma^{-t}(i)}$ and $\mathbf{V}^t_i$ is denoted α^t_i. By convention, objects at time t are the last tracked objects to which an identifier has been assigned. Those at time $t+1$ correspond to the newly detected objects, which must be assigned to the previously identified objects.
Point tracking may either operate data association of new detected points directly on previous recorded points or on estimated points. Point tracking operating on previous recorded is based on the notion of proximity (small position variation) and regular motion (small speed and direction variation). It defines a metric to measure the similarity between the velocity vectors $\mathbf{V}^t$ and the vectors $\mathbf{V}^{t+1}$. With a reasonable capture frequency, the object varies only very slightly in speed and orientation (SETHI et JAIN, 1987). It may be well suited for tracking cars as they do not vary drastically of motion due to their inertia.
Point tracking operating on estimated points seek to predict the position of the points Z^{t+1} denotes $\hat{Z}^{t+1}$, and the associated velocity vectors $\hat{\mathbf{V}}^{t+1}$. In this case, the distance is measured between the vectors $\mathbf{V}^{t+1}$ and $\hat{\mathbf{V}}^{t+1}$.
After measuring a potential match score between two objects, the tracking uses a combinatorial optimization algorithm to establish the best possible average correspondence between all the objects tracked at time t and all the objects detected at time $t+1$, based on the match scores.

B.3.2 Distance for data association

The reference method in this category is carried by Sethi and Jain (SETHI et JAIN, 1987). They define standardized distances which favor the uniformity of directions ($d1^t_{i,j}$) and velocities ($d2^t_{i,j}$), between the vectors $\mathbf{V}^t_i$ and $\mathbf{V}^{t+1}_k$.

$$d1^t_{i,j} = 1 - \frac{\langle \mathbf{V}^t_i, \mathbf{V}^{t+1}_j \rangle}{\|\mathbf{V}^t_i\| \left\|\mathbf{V}^{t+1}_j\right\|} = 1 - cos\left(\mathbf{V}^t_i, \mathbf{V}^{t+1}_j\right) \tag{B.1}$$

The distance $d1_{i,j}$ (equation B.1)is called directional coherence, it can be seen as the cosine of the angle between motion vectors $\mathbf{V}^t_i$ and $\mathbf{V}^{t+1}_j$. The distance is minimal when the two vectors are collinear and in the same direction. As the distance depends on the cosine of the angle, it does not discriminate easily two vectors with a low angle.

$$d2_{i,j}^{t} = 1 - \frac{2 * \sqrt{\|\mathbf{V}_i^t\| \left\|\mathbf{V}_j^{t+1}\right\|}}{\|\mathbf{V}_i^t\| + \left\|\mathbf{V}_j^{t+1}\right\|} = \frac{\left(\sqrt{\|\mathbf{V}_i^t\|} - \sqrt{\left\|\mathbf{V}_j^{t+1}\right\|}\right)^2}{\|\mathbf{V}_i^t\| + \left\|\mathbf{V}_j^{t+1}\right\|} \tag{B.2}$$

The distance $d2_{i,j}$ (equation B.2) is called speed coherence. This term measures the variation of magnitude between the two vectors $\mathbf{V}_i^t$ and $\mathbf{V}_j^{t+1}$. This distance is null when the two vectors magnitudes are equal and it increases as the magnitude differ from each other (Figure B.4). The maximum value is 1. It happens either if $\left\|\mathbf{V}_j^{t+1}\right\|$ or $\|\mathbf{V}_i^t\|$ is null, or if $\left\|\mathbf{V}_j^{t+1}\right\|$ or $\|\mathbf{V}_i^t\|$ is infinite.

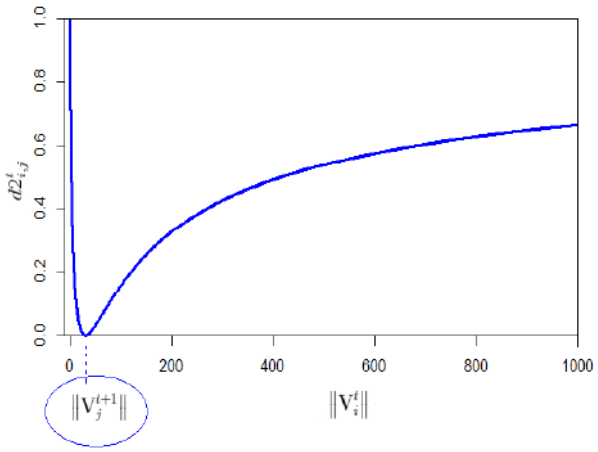

FIGURE B.4 – Curve corresponding to the speed coherence equation for a fixed value of $\left\|\mathbf{V}_j^{t+1}\right\|$. Here $\left\|\mathbf{V}_j^{t+1}\right\| = 30$

This speed coherence does not imply that the difference between the two speeds has to be minimal. If a broiler progress at low speed, this speed coherence might prefer to associate this broiler to far away chicken in the next frame than with itself. This situation appears in particular when broilers stops or get start. Sethi and Jain then define $\Delta = w1 * d1_{i,j}^t + w2 * d2_{i,j}^t$ which is a combination of two distances. The weights w1 and w2 are chosen between 0 and 1, such that $w1 + w2 = 1$.
A second much more classical method is carried by Rangarajan (RANGARAJAN et SHAH, 1991). Its distance is made up of two terms. The first one measures the difference between the velocity vectors formed at times t and $t+1$. Like the Sethi and Jain method, it assumes that speed and direction vary slightly between two consecutive frames. The second one corresponds to speed intensities at time $t+1$. It favors small displacements (low speeds) between two consecutive frames.

$$d3_{i,j}^{t} = \frac{\left\|\mathbf{V}_i^t - \mathbf{V}_{i,j}^{t+1}\right\|}{\sum_{k=1}^{N^t}\sum_{h=1}^{N^{t+1}}\left\|\mathbf{V}_k^t - \mathbf{V}_{k,h}^{t+1}\right\|} + \frac{\left\|\mathbf{V}_{i,j}^{t+1}\right\|}{\sum_{k=1}^{N^t}\sum_{h=1}^{N^{t+1}}\left\|\mathbf{V}_{k,h}^{t+1}\right\|} \tag{B.3}$$

A critical point of these methods comes from the fact that they only depend on the last object's movement. The two main drawbacks are : some movements may require more than one registered movement due to a non straight motion, and a single detection error quickly makes this method unusable. Actually, dealing with more registered points helps to smooth the movement and so to keep direction movement information whereas few past points have been lost.
Rather than measuring the distance between two positions in two successive frames, it could be more advisable to measure this distance with predicted positions computed with more past frames. By using our knowledge on past positions, we can estimate the next objects positions and then we can deal with shortest distances. Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019) defines a normalized displacement estimation. The displacement is equal to 0 if the difference between the predicted position and the estimated position is null. The displacement is equal to 1 if this difference is greater or equal to the object's size. The object's size is written n_{dst}. It is deduced from the dimensions of the region circumscribed to the object, resulting from the detection. It implies that between two images, the object's displacement is less than its size. The definition of the distance between the position of the object at Z_i^{t+1} and its estimated position $\hat{Z}_i^{t+1}$ is written :

$$d4_i^t = min\left(1, \frac{\left\| Z_i^{t+1} - \hat{Z}_i^{t+1} \right\|}{n_{dst}}\right) \tag{B.4}$$

The estimation of $\hat{Z}_i^{t+1}$ is based on the following model :

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \gamma_0 \mathbf{V}_i^t + \sum_{s=1}^{P} \gamma_s \mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} + \epsilon_i^t \tag{B.5}$$

Where $\epsilon_i \sim \mathcal{N}(0, \Sigma)$
Unfortunately, the way the author estimated the α_S coefficients in equation B.5 is not developed in (KARUNASEKERA, WANG et ZHANG, 2019). He uses integer values with a higher weight for most recent velocity vectors. The new position estimation is written as follow :

$$\hat{Z}_i^{t+1} = Z_i^t + \hat{\mathbf{V}}_i^{t+1} \tag{B.6}$$

The new velocity vector $\hat{\mathbf{V}}_i^{t+1}$ modeling is defined as :

$$\hat{\mathbf{V}}_i{}^{t+1} = \frac{\sum_{a=1}^{P} a \mathbf{V}^{t-a+1}_{\sigma^{-(t-a),-t}(i)}}{M(M+1)/2} \tag{B.7}$$

P is set to limit the number of past velocity vectors to use. It assumes that only recent velocity vectors are relevant to the new position estimation. According to the author, the optimal value of M is 5, as it is : "*large enough to get good*

average for predictions and small enough not to drift because of old information" (ref : (KARUNASEKERA, WANG et ZHANG, 2019)). We could regret a lack of information about the way to set the different parameters : the parameter M as it may depend on many factors (number of frame per second, etc ...), and the coefficients α_i as it limits the model of the equation B.5 to a particular case. We will see later a more global approach.
Fletcher (FLETCHER, WARWICK et MITCHELL, 1991) defines a method whose parameters are set by the method of Recursive Least Squares. The model can be expressed as follow :

$$Z_i^t = \Psi^T \mathbf{x}_i(t) + \epsilon_i^t \tag{B.8}$$

Where
— $\epsilon \sim \mathcal{N}(0, \Sigma)$
— $\mathbf{x}(t) = (Z^{t-1}_{\sigma^{-t}(i)}, \dots, Z^{t-p}_{\sigma^{-t+p-1}\circ\dots\circ\sigma^{-t}(i)})$ is the set of past broiler positions.
— Ψ^T the $\mathcal{R}^p$ parameters vector to be estimated
Z_i^t is then a linear combination of past broilers positions. The real value of Ψ is unknown. The method seeks to estimate $\hat{\Psi}$ which is a estimator of Ψ. The vector of parameters $\hat{\Psi}$ is recursively updated based on previous estimation errors.
The error of estimation is noted :

$$e(t) = Z_i^t - \hat{\Psi}(t)^T \mathbf{x}(t) \tag{B.9}$$

Where $\hat{\Psi}(t)$ is estimated by :

$$\hat{\Psi}(t) = \hat{\Psi}(t-1) + \mathbf{K}(t)e(t) \tag{B.10}$$

The vector $\mathbf{K}(t)$ is a set of coefficients that manages the speed of convergence of $\hat{\Psi}(t)$ to Ψ. Having small coefficients implies a slow response to update parameters $\hat{\Psi}(t)$ but it is less sensitive to noise. We can find in (GODFREY et JONES, 1986) and (**bozic1979digital**) different ways of fixing $\mathbf{K}(t)$ based on the error covariance matrix.
This method has the advantage to set automatically its parameters unlike the Karanusekara one. On the other hand, this method measures positions and not movements, which makes the estimation position-dependent. The further these positions are from the origin of the image, the greater the variation of the estimated point. To get around the problem, it is judicious to replace positions by velocity vectors. Velocity vectors do not vary as they get away from the origin of the image.

B.4 Prediction models

The method we propose relies on estimating several hundred chickens' movements to predict their future positions. The peculiarity here comes from the fact that, unlike a car or a pedestrian, a chicken does not move at a constant

speed with a well-defined direction. We therefore seek to determine an estimator $\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)}$ of $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$ such that $\epsilon_i = \left\| \hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} - \mathbf{V}^{T+1}_{\sigma^{t+1}(i)} \right\|$ has the smallest variance. Since hundreds of chickens may create many occlusions, the movement predictor has to deal with low detection rate. Finally, as the model of detection may not be perfect, the method has to deal with motion due to detection imprecision. As the chickens position may vary slightly around their actual positions, the movement estimation method has to take it into account to determine the natural movement of chickens.
Given what has already been done, the new method must meet specific requirements.

— It must be adapted to the type of object followed to be as precise as possible.
— This method must be independent of the object's activity. The parameters must therefore be robust whatever the activity is.
— The method must be independent of the displacement orientation. Rotating the camera should not influence the results.
— The coefficients used should ideally be known from the start of the tracking to manage the incoming objects more easily and limit the calculation times during the tracking.

B.4.1 Methodology

In order to estimate the parameters required by the different methods, we dispose of several tracking data sets of reference whose points position have been manually identified. The results have been checked experimentally. When an object seems to disappear from the video due to occlusions or poor detection, its velocity vector was set to null vector as we did not have any information about the new displacement.

B.4.1.1 The models

First model The first model consists in considering the movement of the animal as being null.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \epsilon^t_i \tag{B.11}$$

With : $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$. This model can be considered as a reference one. The matrix ϵ is diagonal, and its terms are inversely proportional to the squares of the pixel area.

Second model The second model is based on the notion of constant motion from Sethi and Jain (section B.3.2). It seeks for two same movements in two consecutive frames.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \mathbf{V}^t_i + \epsilon^t_i \tag{B.12}$$

With : $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$.

This model is well suited when the frame rate is high and motion varies slowly between two consecutive frames.

Third model This model is the Hasith Karunasekera (KARUNASEKERA, WANG et ZHANG, 2019) one.

$$\mathbf{V}^{t+1}_{\sigma^{t+1}(i)} = \gamma_0 \mathbf{V}^t_i + \sum_{s=1}^{P} \gamma_s \mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} + \epsilon^t_i \tag{B.13}$$

Where $\epsilon^t_i \sim \mathcal{N}(0, \Sigma)$ and $\gamma_s \in \mathbb{R}$, with $0 \leq \gamma_s < 1$

B.4.1.2 Motion prediction

Based on these three models, it is possible to define different motion predictions.

Nearest Neighbor This method predicts movement based on the model B.12. There are no parameters to estimate.

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = 0 \tag{B.14}$$

The estimated position $\hat{Z}^{t+1}_{\sigma^{t+1}(i)}$ is therefore in a region centered in Z^t_i and whose radius depends on the variance of $\mathbf{V}^{t+1}_{\sigma^{t+1}(i)}$.

Regular motion The movement prediction is based on the model B.15. It does not need any parameters to set either.

$$\hat{Z}^{t+1}_{\sigma^{t+1}(i)} = Z^t_i + \hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} \tag{B.15}$$

In such situation the predicted position is estimated as :

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \mathbf{V}^t_i \tag{B.16}$$

Weighted Average This one is based on the model B.13. The parameters are set as in the same they are presented in the article (KARUNASEKERA, WANG et ZHANG, 2019) where the model comes from. P is fixed to 5 and $\gamma_s \in 1, 2, 3, 4, 5$.

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \frac{1}{3}\mathbf{V}^t_i + \frac{1}{15}\sum_{s=1}^{4}(5-s)\mathbf{V}^{t-s}_{\sigma^{-(t-s+1),-t}(i)} \tag{B.17}$$

Cartesian Least Squares It is the same model as the Karunasekera one (B.13), but differs by the way of estimating the model parameters. We estimate the parameters M and γ_s, based on reference data sets. This time the estimated position is written as :

$b = u + iv \in \mathcal{R}^m$ is the output representing current movement values.

$$b = u + iv = \begin{pmatrix} \mathbf{V}_{X_1}^t \\ \vdots \\ \mathbf{V}_{X_m}^t \end{pmatrix} + i \begin{pmatrix} \mathbf{V}_{Y_1}^t \\ \vdots \\ \mathbf{V}_{Y_m}^t \end{pmatrix}$$

$z = x + iy \in \mathcal{R}^n$ is the vector of parameters to estimate.

$$z = x + iy = Re \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{pmatrix} + iIm \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_n \end{pmatrix}$$

The solution of γ is written under the form :

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} BB^T + CC^T & -C^TB + B^TC \\ C^TB - B^TC & BB^T + CC^T \end{pmatrix}^{-1} \begin{pmatrix} B^T & C^T \\ C^T & B^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{B.20}$$

Polar Least Squares This method is based on the model B.13. It is the equivalent of the method B.4.1.2 but in polar coordinates, with $\mathbf{V}^t = (R^t, \theta^t)$. An angle may represent either a direction or a rotation. While it makes sense to sum rotation angles (two rotations of $\pi/4$ is equivalent to one rotation of $\pi/2$), the summation of $\pi/2$ direction with a $-\pi/2$ direction has no physical interpretation. This is why it is better to deal with variation of orientation (rotation) referred as $\Delta\Theta$ in Figure B.3 and not directly with velocity vector directions Θ. Dealing with polar coordinates faces to constraints on speeds. In this sense, speeds values R are forced to be positives, as variations of orientation are includes in $[-\pi, \pi)$.

To ensure a positive speed value, the analysis uses the logarithm of the speed value.

$$\begin{pmatrix} log(R_1^t) \\ \vdots \\ log(R_m^t) \end{pmatrix} = \begin{pmatrix} log(R_1^{t-1}) & \dots & log(R_1^{t-n}) \\ \vdots & & \vdots \\ log(R_1^{t-1}) & \dots & log(R_1^{t-n}) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}$$

The motion is then estimated as :

$$R_i^t = \prod_{s=1}^{t-1} \left(R^{t-s}_{\sigma^{-(t-s+1),-t}(i)} \right)^{\alpha_s} \tag{B.21}$$

$$\Delta\Theta_i^t = \sum_{s=1}^{t-1} \beta_s \Delta\Theta^{t-s}_{\sigma^{-(t-s+1),-t}(i)} \tag{B.22}$$

$$\Delta\Theta_i^{t-1} = \Theta_i^t - \Theta_i^{t-1} \tag{B.23}$$

$$\hat{\mathbf{V}}^{t+1}_{\sigma^{t+1}(i)} = \hat{R}^{t+1}_{\sigma^{t+1}(i)} \begin{pmatrix} cos\left(\Theta_i^t + \Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)}\right) \\ sin\left(\Theta_i^t + \Delta\hat{\Theta}^{t+1}_{\sigma^{t+1}(i)}\right) \end{pmatrix} \tag{B.24}$$

Parameters in B.21 and B.22 are also estimated with least squares method on the reference database.
As said previously, a non detected object would be considered to be at the same position as its last measure, since its motion is set as null. It means, a null speed but also an undetermined orientation of motion. Because *undetermined* is a number difficult to handle with, we arbitrary set it to zero.

Least Squares Under Constraints In the previous motion estimation there is no dependence between speed values and variations of orientation. Thanks to the tracking database created manually (all objects are detected, there are no false positive and no tracking error), it is possible to check the relationship between the speed and the variation of orientation. Variations of orientation are centered around zero. For each cell of the 2 dimensional histogram in the Figure B.5, the intensity represents the occurrence of couples speed and orientation variation. It can be noticed that for small speed the range of orientation variation is large. This range decreases as speed increases. It means that a displacement of 40 pixels/image paired with a π orientation variation is considered as an outlier. Based on the maximum recorded speed, two red lines are drawn as a first order approximation of the limit of the distribution. The idea is not to model exactly the relationship between speeds and variations of orientation, but to limit outliers.
High speed values are therefore accepted when the expected orientation variation is small. On the other hand, if the speed estimator is high, constraints are applied on variation orientation in order to limit outliers. This approximation of the distribution limit put constraints to the speed and to the orientation variation estimation of the model **??**.

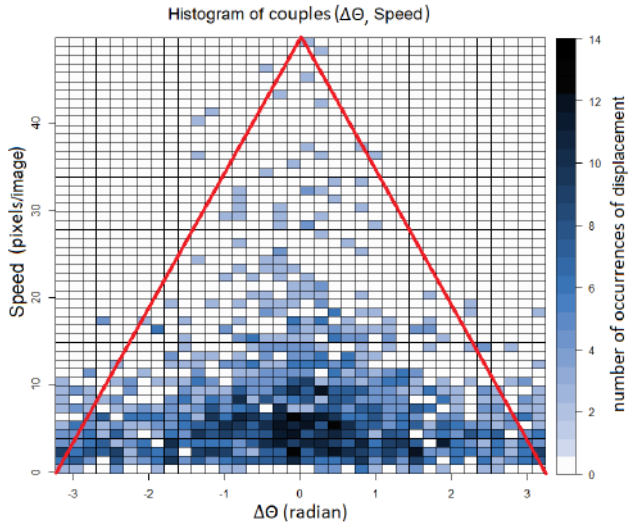

FIGURE B.5 – Relation between the speed and the variation of orientation

Once $\hat{R}_i^t$ and $\Delta\hat{\Theta}_i^t$ are estimated they are used to constrain new estimators $\hat{R'}_i^t$ and $\Delta\hat{\Theta'}_i^t$

$$\hat{R'}_i^t = \hat{R}_i^t \left(1 - abs\left(\frac{\Delta\hat{\Theta}_i^t}{\pi}\right)\right) \tag{B.25}$$

$$\Delta\hat{\Theta'}_i^t = min\left(\Delta\hat{\Theta}_i^t, \pi\left(1 - \frac{\hat{R}_i^t}{max(R)}\right)\right) \tag{B.26}$$

The value $max(R)$ is measured from the data base of reference.
The estimation is then written as :

$$\hat{Z}_{\sigma^{t+1}(i)}^{t+1} = Z_i^t + \hat{\mathbf{V}}_{\sigma^{t+1}(i)}^{t+1} \tag{B.27}$$

$$\hat{\mathbf{V}}_{\sigma^{t+1}(i)}^{t+1} = \hat{R'}_{\sigma^{t+1}(i)}^{t+1} \begin{pmatrix} cos\left(\Theta_i^t + \Delta\hat{\Theta'}_{\sigma^{t+1}(i)}^{t+1}\right) \\ sin\left(\Theta_i^t + \Delta\hat{\Theta'}_{\sigma^{t+1}(i)}^{t+1}\right) \end{pmatrix} \tag{B.28}$$

B.4.2 Videos and detections

Three videos where used to estimate the models parameters.

When broilers grow up, the rate of detected broilers over total number of broilers increases. Here, the activity is referred as the global motion of broilers. The more broilers are moving with high speeds the higher is the activity. High speed motion is generally accompanied by flapping wings. A false positive is detection recorded where no broiler is present. The minimal recorded distance between broilers is a threshold below which two broilers positions

	21 days of age	26 days of age	37 days of age
Surface (m^2)	3.4	3.8	3.8
Mean number of broilers	34	64	38
Activity	high	low	low
Speed $(pixel.s^{-1})$	826	630	544
Distance intra broilers (pixel)	36.9	32.5	34.4
Broilers size	106	102	141
Sensitivity	97	99	99
Precision	0.2	2.28	1.6

TABLE B.1 – Specifications of each video. The first row is the camera field area. The second row is the mean number of broilers over each image of the video. The third row is the broilers activity intensity. The fourth row is the maximal recorded speed in the video. The fifth row is the minimal distance recorded between the closest broilers. The sixth row is the broilers largest size median. The seventh row is percentage of detected boilers out of the CNN. The last row is the percentage of false positive out the CNN

can not be recorded.

Finally, for each broiler is given its bounding box size. The last row of the Table **??** gives the median of the larger bounding box size of all broilers. The characteristics of the three videos (see Figures B.6, B.7, B.8) used to estimate the model parameters are summarized in Table B.1. They differ mainly in the age of broilers, in their activity and density (see Table B.1).
The first video corresponds to a twenty one days of age broilers flock. This is the video with the higher activity. It is the video with the lower rate of detection, only 97% of present broilers are detected.
Broilers of the next video are 26 days of age. This second video corresponds to the most tightly spaced broilers. Broilers trying to clear a path in such crowded area may hide other broilers and cause tracking errors. The last video corresponds to 37 days of age broilers. Broilers are much bigger compare to those in the two previous videos. Even though the density and the activity are quite low, occlusions might occur due to their biggest size. This being said, this video records the less interactions between broilers compare to the two previous videos.

B.4.3 Parameters estimations

Two data sets are associated to each video. The first data set is the reference one. A surrounding bounding box has been drawn manually around all broilers in order to have a good localization, without any miss detection nor false positive detection. The second data set are data generated by a convolutional neural network. These data are used to test the tracking motion estimation methods. Tracking has been run over the reference data set and then over the

FIGURE B.6 – 21 days of age

FIGURE B.7 – 26 days of age

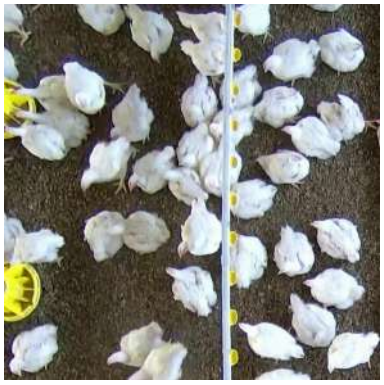

FIGURE B.8 – 37 days of age

test data set for the six following tracking methods which differs by the way the motion is estimated (section B.4.1.2) :
— Nearest Neighbor (Method 1)
— Regular motion (Method 2)
— Weighted Average (Method 3)
— Cartesian Least Squares (Method 4)
— Polar Least Squares (Method 5)
— Least Squares Under Constraints (Method 6)
The objective is to have common coefficients for all data sets. A method has to be robust, to the density and to the broiler activities. As a first step, coefficients have been evaluated for each data set of reference corresponding to broilers excursively in movement. As most part of them are generally resting, the idea is to avoid to have parameters biased by too much resting broilers. The objective being to predict motion. The Table B.2 gives these parameters estimations given by the least squares method. An analyze has been initiated in order to determine the parameter P in the model B.13, which is the number of past relevant velocity vectors. By estimating these γ_s parameters, it turned out that they are higher as recorded measures are more recent. The minimum parameter value accepted is then defined such that the model does not take into account value whose contribution leads to a movement lower to one pixel for highest speed values.

TABLE B.2 – Least square parameters estimation

Model	Parameters	Day 21	Day 26	Day 37
Cartesian	$\hat{\gamma}$	(0.43 0.31)	(0.33 0.25)	(0.27 0.17)
Polar	$\begin{pmatrix}\hat{\alpha}\\ \hat{\beta}\end{pmatrix}$	$\begin{pmatrix}0.38 & 0.28 & 0.17\\ -0.07 & -0.09 & -0.07\end{pmatrix}$	$\begin{pmatrix}0.35 & 0.27 & 0.21\\ -0.05 & 0.03 & -0.02\end{pmatrix}$	$\begin{pmatrix}0.34 & 0.27 & 0.18\\ -0.06 & -0.04 & -0.007\end{pmatrix}$

Based on these results, the final parameters used for the test data sets are the mean of the three reference data sets. These estimations have not been estimated on a whole data set merging those three ones in order to avoid that one of them prevail over the two others.

$$\hat{\gamma} = \begin{pmatrix} 0.34 \\ 0.24 \end{pmatrix} \tag{B.29}$$

$$\begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} = \begin{pmatrix} 0.36 & 0.27 & 0.19 \\ -0.06 & -0.03 & -0.03 \end{pmatrix} \tag{B.30}$$

Table B.3 gives then a global information on the movement estimation. This test consists in measuring the standard deviation of $\hat{\epsilon} = \hat{\mathbf{V}}^t - \mathbf{V}^T$ for each model. Smaller is the variance values, better is the model prediction. It has to be noticed that it is based on reference data sets with broilers in movement. The method 1 (Nearest Neighbor) seems to be the less adapted one to broilers high speeds and the hybrid the most adapted one. However, in data test most of broilers are resting and detection is not perfect.

TABLE B.3 – Standard deviation of the absolute difference between measured and estimated positions according to the motion estimation method

Model	Day 21	Day 26	Day 37
Method 1	13.34	6.71	10.53
Method 2	11.5	7.13	11.21
Method 3	10.26	5.75	8.8
Method 4	9.47	5.77	8.81
Method 5	6.82	4.18	5.87
Method 6	5.47	3.47	4.48

B.5 Results and discussion

One last parameter to take into account is the maximal speed reach by broilers. The fact is that if the tracking leads to wrong detections, wrong identification assignments occurs. They may appear between false positives and miss detected or occluded broilers. There is a compromise to be find between to enable large speed displacements in order to track fast broilers, and to enable only low speed displacement in order to avoid wrong assignments with occluded and miss detected broilers. Testing all the six methods on the reference data sets (no occlusion nor miss detection) may give an idea of the highest speed displacement that can be accepted. The figure B.5 gives the number of tracking error according to the threshold of largest accepted speed. This threshold is expressed as a ratio of broilers size. A value of 100% refers to the median of the larger bounding box size of every broilers (cf : Table **??**). The best method is the one which reaches the minimum of error for the smallest speed threshold possible. Being able to track broilers with a small

speed threshold express a good estimation of motion and leads to less wrong identification assignments. Where it might be possible to correct errors due to too high speeds in post processing, it is quite impossible to correct wrong assignment identifications for ID passing from one broiler to another due to miss detections. This is why it is really important to reach the minimum of error as quickly as possible.

B.5.1 Analyze of performance for each method

Method 1 Nearest Neighbor As expected Nearest Neighbor method is one of the worsts as predict high speed motions. Even with the reference data (perfect detection), it is one of the last to reach a minimum of errors. Its best performance is found for high density video in test situation.

Method 2 Regular motion based on (SETHI et JAIN, 1987) (section **??**), it was expected to have good performances for straight movements and good quality of detection. Straight movements are mainly presents in the video of 21 days of age broilers as speeds are the higher. In practice its best performance is effectively find the the 21 days of age broilers data test where it reaches quickly the minimum of errors. However performances decrease for low broiler activities both in reference and test data sets.

Method 3 Weighted Average the Weighted Average method from (KARUNASEKERA, WANG et ZHANG, 2019) was expected to be robust to miss detection as it uses up to five input variables, however it was said it probably does not use the most adapted coefficients. When it is compared to other methods, it seems to have better performance for reference data set compare to test data test. That means it is dependent on the detection quality.

Method 4 Cartesian Least Squares In theory, this method should follow the WA method performance as they are based on the same model. However, it coefficients have been tuned on reference data and it uses less coefficients making it more sensitive to poor detection. In practice, the XY method is always among the two fastest methods to reach the minimum of error. It looks like the best compromise according to the three situations.

Method 5 & 6 Polar Least Squares & Least Squares Under Constraints Even though their coefficients have been tuned with reference data set, the Polar method remain one of the worst one both for reference and test data sets. However these methods did present the lowest standard deviation in Table B.3. It means they are too sensitive to movements due to imprecision of detection. Nevertheless, the limit of outliers by the Hybrid method seems really efficient when the two methods are compared together. Even though the Hybrid method is quite slow to reach the minimum for low densities video (21 and

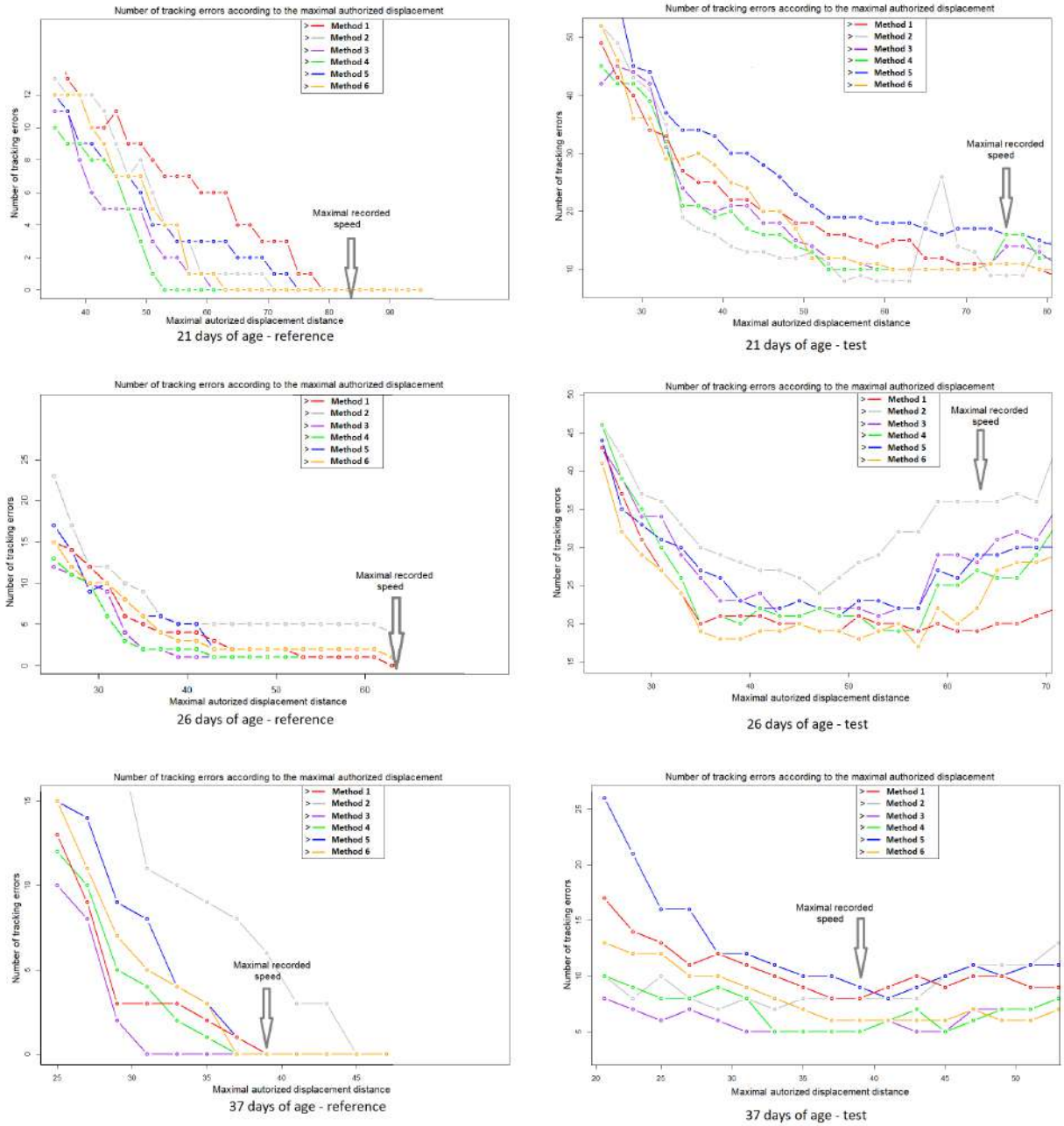

FIGURE B.9 – Number of tracking error according to the authorized displacement distance

37 days of age), it is among those which reach the lowest number of tracking errors and seems to reach the best performance for high density video.
A successful method of motion prediction has to be robust to : the density of the broilers, to their activities, to motions derived from the detection imprecision, to the occlusions and miss detections. In addition, it has to reach the best performances for the lowest threshold of authorized speed motion. In this sens the XY method seems to be the best compromise among all the six methods investigated.

B.6 Conclusion

In this paper, three estimations of motion applied to broilers tracking have been presented. They have been compared to three methods of the literature through three data sets created by our own. The XY method stands out of the group as the best adapted to all situations. However, even though one of them seems to be most adapted to all kind of situation, it is still dependent of maximum authorized speed threshold that seems to vary a lot according to the video. An automatic way to fix max displacement has to be set in order to have a fully robust method.

Bibliographie

ABDELMOUNAIME, Safia et He DONG-CHEN (2013). « New Brodatz-based image databases for grayscale color and multiband texture analysis ». In : *International Scholarly Research Notices* 2013.

AMRAEI, Somaye, Saman ABDANAN MEHDIZADEH et Somayeh SALLARY (2017). « Application of computer vision and support vector regression for weight prediction of live broiler chicken ». In : *Engineering in Agriculture, Environment and Food* 10.4, p. 266 -271. ISSN : 1881-8366. DOI : https://doi.org/10.1016/j.eaef.2017.04.003. URL : http://www.sciencedirect.com/science/article/pii/S1881836617301076.

BALAJI, S. R. et S. KARTHIKEYAN (2017). « A survey on moving object tracking using image processing ». In : *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, p. 469-474. DOI : 10.1109/ISCO.2017.7856037.

BALLARD, Dana H (1981). « Generalizing the Hough transform to detect arbitrary shapes ». In : *Pattern recognition* 13.2, p. 111-122.

BAR-SHALOM, Yaakov, Fred DAUM et Jim HUANG (2009). « The probabilistic data association filter ». In : *IEEE Control Systems Magazine* 29.6, p. 82-100.

BARRETO, Joao et al. (2009). « Automatic camera calibration applied to medical endoscopy ». In : *BMVC 2009-20th British Machine Vision Conference*. The British Machine Vision Association (BMVA), p. 1-10.

BAY, Herbert, Tinne TUYTELAARS et Luc VAN GOOL (2006). « Surf : Speeded up robust features ». In : *European conference on computer vision*. Springer, p. 404-417.

BENESTY, Jacob et al. (2009). « Pearson correlation coefficient ». In : *Noise reduction in speech processing*. Springer, p. 1-4.

BENGIO, Yoshua (2009). *Learning deep architectures for AI*. Now Publishers Inc.

BERCLAZ, Jerome, Francois FLEURET et Pascal FUA (2006). « Robust people tracking with global trajectory optimization ». In : *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. T. 1. IEEE, p. 744-750.

BERTALMIO, M., G. SAPIRO et G. RANDALL (2000). « Morphing active contours ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.7, p. 733-737. DOI : 10.1109/34.865191.

BI, Fangming et al. (jan. 2019). « Review on Video Object Tracking Based on Deep Learning ». In : *Journal of New Media* 1, p. 63-74. DOI : 10.32604/jnm.2019.06253.

BIRCHFIELD, Stan (1998). « Elliptical head tracking using intensity gradients and color histograms ». In : *Proceedings. 1998 IEEE Computer Society conference on computer vision and pattern recognition (Cat. No. 98CB36231)*. IEEE, p. 232-237.

BLACKMAN, Samuel S (2004). « Multiple hypothesis tracking for multiple target tracking ». In : *IEEE Aerospace and Electronic Systems Magazine* 19.1, p. 5-18.

BLOOM, Stephen A (1981). « Similarity indices in community studies : potential pitfalls ». In : *Marine Ecology Progress Series* 5.2, p. 125-128.

BLUM, Harry et al. (1967). *A transformation for extracting new descriptors of shape*. T. 43. MIT press Cambridge, MA.

BOCHINSKI, Erik, Volker EISELEIN et Thomas SIKORA (2017). « High-speed tracking-by-detection without using image information ». In : *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, p. 1-6.

BREIMAN, Leo et al. (2017). *Classification and regression trees*. Routledge.

BREITENSTEIN, Michael D et al. (2009). « Robust tracking-by-detection using a detector confidence particle filter ». In : *2009 IEEE 12th International Conference on Computer Vision*. IEEE, p. 1515-1522.

BRENDEL, William, Mohamed AMER et Sinisa TODOROVIC (2011). « Multiobject tracking as maximum weight independent set ». In : *CVPR 2011*. IEEE, p. 1273-1280.

CANNY, John (1986). « A computational approach to edge detection ». In : *IEEE Transactions on pattern analysis and machine intelligence* 6, p. 679-698.

CARRANZA-GARCÍA, Manuel et al. (2021). « On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data ». In : *Remote Sensing* 13.1, p. 89.

CHEN, Hwann-Tzong et Tyng-Luh LIU (2001). « Trust-region methods for real-time tracking ». In : *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. T. 2. IEEE, p. 717-722.

CHENG, Heng-Da et al. (2001). « Color image segmentation : advances and prospects ». In : *Pattern recognition* 34.12, p. 2259-2281.

CHENG, Yizong (1995). « Mean shift, mode seeking, and clustering ». In : *IEEE transactions on pattern analysis and machine intelligence* 17.8, p. 790-799.

CIAPARRONE, Gioele et al. (2020). « Deep learning in video multi-object tracking : A survey ». In : *Neurocomputing* 381, p. 61 -88. ISSN : 0925-2312. DOI : https://doi.org/10.1016/j.neucom.2019.11.023. URL : http://www.sciencedirect.com/science/article/pii/S0925231219315966.

COHEN, Patricia, Stephen G WEST et Leona S AIKEN (2014). *Applied multiple regression/correlation analysis for the behavioral sciences*. Psychology press.

COLLINS, Lisa M. (2008). « Non-intrusive tracking of commercial broiler chickens in situ at different stocking densities ». In : *Applied Animal Behaviour Science* 112.1, p. 94 -105. ISSN : 0168-1591. DOI : https://doi.org/10.1016/j.applanim.2007.08.009. URL : http://www.sciencedirect.com/science/article/pii/S0168159107002626.

COMANICIU, Dorin, Visvanathan RAMESH et Peter MEER (2003). « Kernel-based object tracking ». In : *IEEE Transactions on pattern analysis and machine intelligence* 25.5, p. 564-577.

DEMPSTER, Arthur P, Nan M LAIRD et Donald B RUBIN (1977). « Maximum likelihood from incomplete data via the EM algorithm ». In : *Journal of the Royal Statistical Society : Series B (Methodological)* 39.1, p. 1-22.

DENG, Jia et al. (2009). « Imagenet : A large-scale hierarchical image database ». In : *2009 IEEE conference on computer vision and pattern recognition*. Ieee, p. 248-255.

DHILLON, Anamika et Gyanendra K VERMA (2020). « Convolutional neural network : a review of models, methodologies and applications to object detection ». In : *Progress in Artificial Intelligence* 9.2, p. 85-112.

EICHKITZ, Christoph Georg, Johannes AMTMANN et Marcellus Gregor SCHREILECHNER (2013). « Calculation of grey level co-occurrence matrix-based seismic attributes in three dimensions ». In : *Computers & Geosciences* 60, p. 176-183.

FANG, Cheng et al. (2020). « Comparative study on poultry target tracking algorithms based on a deep regression network ». In : *Biosystems Engineering* 190, p. 176 -183. ISSN : 1537-5110. DOI : https://doi.org/10.1016/j.biosystemseng.2019.12.002. URL : http://www.sciencedirect.com/science/article/pii/S1537511019309092.

FANG, Cheng et al. (2021). « Pose estimation and behavior classification of broiler chickens based on deep neural networks ». In : *Computers and Electronics in Agriculture* 180, p. 105863. ISSN : 0168-1699. DOI : https://doi.org/10.1016/j.compag.2020.105863. URL : http://www.sciencedirect.com/science/article/pii/S0168169920305159.

FEIYANG, Zhang et al. (2016). « Monitoring behavior of poultry based on RFID radio frequency network ». In : *International Journal of Agricultural and Biological Engineering* 9.6, p. 139-147.

FELZENSZWALB, Pedro F et Daniel P HUTTENLOCHER (2004). « Efficient graph-based image segmentation ». In : *International journal of computer vision* 59.2, p. 167-181.

FIX, Evelyn et Joseph Lawson HODGES (1989). « Discriminatory analysis. Nonparametric discrimination : Consistency properties ». In : *International Statistical Review/Revue Internationale de Statistique* 57.3, p. 238-247.

FLETCHER, MJ, Kevin WARWICK et Richard J MITCHELL (1991). « Application of a hybrid tracking algorithm to motion analysis. » In : *CVPR*, p. 84-89.

FORTMANN, Thomas E, Yaakov BAR-SHALOM et Molly SCHEFFE (1980). « Multi-target tracking using joint probabilistic data association ». In : *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*. IEEE, p. 807-812.

FRAGKIADAKI, Katerina et Jianbo SHI (2011). « Detection free tracking : Exploiting motion and topology for segmenting and tracking under entanglement ». In : *CVPR 2011*. IEEE, p. 2073-2080.

FREUND, Yoav, Robert SCHAPIRE et Naoki ABE (1999). « A short introduction to boosting ». In : *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612.

FROMM, Michael et al. (2019). « Automated detection of conifer seedlings in drone imagery using convolutional neural networks ». In : *Remote Sensing* 11.21, p. 2585.

FU, Huazhu et al. (2018). « Joint optic disc and cup segmentation based on multi-label deep network and polar transformation ». In : *IEEE transactions on medical imaging* 37.7, p. 1597-1605.

FUCHS, Fabian B et al. (2019). « End-to-end recurrent multi-object tracking and trajectory prediction with relational reasoning ». In : *arXiv preprint arXiv :1907.12887*.

GALE, David et Lloyd S SHAPLEY (1962). « College admissions and the stability of marriage ». In : *The American Mathematical Monthly* 69.1, p. 9-15.

GARCIA-GARCIA, Alberto et al. (2018). « A survey on deep learning techniques for image and video semantic segmentation ». In : *Applied Soft Computing* 70, p. 41-65.

GIL, Joseph Yossi et Ron KIMMEL (2002). « Efficient dilation, erosion, opening, and closing algorithms ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.12, p. 1606-1617.

GIRSHICK, Ross (2015). « Fast r-cnn ». In : *Proceedings of the IEEE international conference on computer vision*, p. 1440-1448.

GIRSHICK, Ross et al. (2014). « Rich feature hierarchies for accurate object detection and semantic segmentation ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 580-587.

GODFREY, Keith et Peter JONES (1986). *Signal processing for control*. T. 79. Springer.

GUO, Yangyang et al. (2020). « A Machine Vision-Based Method for Monitoring Broiler Chicken Floor Distribution ». In : *Sensors* 20.11. ISSN : 1424-8220. DOI : 10.3390/s20113179. URL : https://www.mdpi.com/1424-8220/20/11/3179.

HAAR, Alfred (1910). « Zur theorie der orthogonalen funktionensysteme ». In : *Mathematische Annalen* 69.3, p. 331-371.

HAFIZ, Abdul Mueed et Ghulam Mohiuddin BHAT (2020). « A survey on instance segmentation : state of the art ». In : *International journal of multimedia information retrieval*, p. 1-19.

HAN, Mei et al. (2004). « A detection-based multiple object tracking method ». In : *2004 International Conference on Image Processing, 2004. ICIP'04.* T. 5. IEEE, p. 3065-3068.

HARALICK, Robert M, Karthikeyan SHANMUGAM et Its' Hak DINSTEIN (1973). « Textural features for image classification ». In : *IEEE Transactions on systems, man, and cybernetics* 6, p. 610-621.

HARRIS, Christopher G, Mike STEPHENS et al. (1988). « A combined corner and edge detector ». In : *Alvey vision conference*. T. 15. 50. Citeseer, p. 10-5244.

HE, Kaiming et al. (2015). « Spatial pyramid pooling in deep convolutional networks for visual recognition ». In : *IEEE transactions on pattern analysis and machine intelligence* 37.9, p. 1904-1916.

— (2016). « Deep residual learning for image recognition ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770-778.

HE, Kaiming et al. (2017). « Mask r-cnn ». In : *Proceedings of the IEEE international conference on computer vision*, p. 2961-2969.

HU, Ming-Kuei (1962). « Visual pattern recognition by moment invariants ». In : *IRE transactions on information theory* 8.2, p. 179-187.

HU, Rui, Mark BARNARD et John COLLOMOSSE (2010). « Gradient field descriptor for sketch based retrieval and localization ». In : *2010 IEEE International conference on image processing*. IEEE, p. 1025-1028.

HUA, Yang, Karteek ALAHARI et Cordelia SCHMID (2015). « Online object tracking with proposal selection ». In : *Proceedings of the IEEE international conference on computer vision*, p. 3092-3100.

HUTTENLOCHER, D. P., J. J. NOH et W. J. RUCKLIDGE (1993). « Tracking non-rigid objects in complex scenes ». In : *1993 (4th) International Conference on Computer Vision*, p. 93-101. DOI : 10.1109/ICCV.1993.378231.

ILLINGWORTH, John et Josef KITTLER (1988). « A survey of the Hough transform ». In : *Computer vision, graphics, and image processing* 44.1, p. 87-116.

ILTIS, Ronald A et P-Y TING (1991). « Data association in multi-target tracking : A solution using a layered Boltzmann machine ». In : *IJCNN-91-Seattle International Joint Conference on Neural Networks*. T. 1. IEEE, p. 31-36.

INTILLE, S. S., J. W. DAVIS et A. F. BOBICK (1997). « Real-time closed-world tracking ». In : *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 697-703. DOI : 10.1109/CVPR.1997.609402.

JIANG, Xiaolong et al. (2019). « Graph neural based end-to-end data association framework for online multiple-object tracking ». In : *arXiv preprint arXiv :1907.05315*.

JOSHI, Kinjal A et Darshak G THAKORE (2012). « A survey on moving object detection and tracking in video surveillance system ». In : *International Journal of Soft Computing and Engineering* 2.3, p. 44-48.

JU, S. et al. (2020). « Video Tracking to Monitor Turkey Welfare ». In : *2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, p. 50-53. DOI : 10.1109/SSIAI49293.2020.9094604.

KARUNASEKERA, Hasith, Han WANG et Handuo ZHANG (2019). « Multiple object tracking with attention to appearance, structure, motion and size ». In : *IEEE Access* 7, p. 104423-104434.

KASS, Michael, Andrew WITKIN et Demetri TERZOPOULOS (1988). « Snakes : Active contour models ». In : *International journal of computer vision* 1.4, p. 321-331.

KIM[1], JINHO, BS KIM et Silvio SAVARESE (2012). « Comparing image classification methods : K-nearest-neighbor and support-vector-machines ». In : *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*. T. 1001, p. 48109-2122.

KOTHIYA, Shraddha et Kinjal MISTREE (jan. 2015). « A review on real time object tracking in video sequences ». In : p. 1-4. DOI : 10.1109/EESCO.2015.7253705.

KUHN, Harold W et Albert W TUCKER (2014). « Nonlinear programming ». In : *Traces and emergence of nonlinear programming*. Springer, p. 247-258.

LANCE, Godfrey N et William T WILLIAMS (1966). « Computer programs for hierarchical polythetic classification ("similarity analyses") ». In : *The Computer Journal* 9.1, p. 60-64.

LAUNEAU, PATRICK et P-YF ROBIN (1996). « Fabric analysis using the intercept method ». In : *Tectonophysics* 267.1-4, p. 91-119.

LECUN, Yann et al. (1989). « Backpropagation applied to handwritten zip code recognition ». In : *Neural computation* 1.4, p. 541-551.

LI, Wei et al. (2021). « Chicken image segmentation via multi-scale attention-based deep convolutional neural network ». In : *IEEE Access* 9, p. 61398-61407.

LI, Xin et al. (2010). « A multiple object tracking method using Kalman filter ». In : *The 2010 IEEE international conference on information and automation*. IEEE, p. 1862-1866.

LIN, Liang et al. (2015). « Detection-free multiobject tracking by reconfigurable inference with bundle representations ». In : *IEEE transactions on cybernetics* 46.11, p. 2447-2458.

LIN, Tsung-Yi et al. (2014). « Microsoft coco : Common objects in context ». In : *European conference on computer vision*. Springer, p. 740-755.

LIN, Tsung-Yi et al. (2017). « Feature pyramid networks for object detection ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2117-2125.

LOWE, David G (1999). « Object recognition from local scale-invariant features ». In : *Proceedings of the seventh IEEE international conference on computer vision*. T. 2. Ieee, p. 1150-1157.

— (2004). « Distinctive image features from scale-invariant keypoints ». In : *International journal of computer vision* 60.2, p. 91-110.

LU, Dengsheng et Qihao WENG (2007). « A survey of image classification methods and techniques for improving classification performance ». In : *International journal of Remote sensing* 28.5, p. 823-870.

LUCAS, Bruce D, Takeo KANADE et al. (1981). « An iterative image registration technique with an application to stereo vision ». In : Vancouver, British Columbia.

LUITEN, Jonathon, Idil Esen ZULFIKAR et Bastian LEIBE (2020). « Unovost : Unsupervised offline video object segmentation and tracking ». In : *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, p. 2000-2009.

MACQUEEN, James et al. (1967). « Some methods for classification and analysis of multivariate observations ». In : *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. T. 1. 14. Oakland, CA, USA, p. 281-297.

MARR, David et Ellen HILDRETH (1980). « Theory of edge detection ». In : *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207.1167, p. 187-217.

MATAS, Jiri et al. (2004). « Robust wide-baseline stereo from maximally stable extremal regions ». In : *Image and vision computing* 22.10, p. 761-767.

MCCULLOCH, Warren S et Walter PITTS (1943). « A logical calculus of the ideas immanent in nervous activity ». In : *The bulletin of mathematical biophysics* 5.4, p. 115-133.

MIKOLAJCZYK, Krystian et Cordelia SCHMID (2005). « A performance evaluation of local descriptors ». In : *IEEE transactions on pattern analysis and machine intelligence* 27.10, p. 1615-1630.

MITTAL, Sparsh (2019). « A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform ». In : *Journal of Systems Architecture* 97, p. 428-442.

MITTAL, Usha, Sonal SRIVASTAVA et Priyanka CHAWLA (2019). « Review of different techniques for object detection using deep learning ». In : *Proceedings of the Third International Conference on Advanced Informatics for Computing Research*, p. 1-8.

MOLLAH, Md. Bazlur R. et al. (2010). « Digital image analysis to estimate the live weight of broiler ». In : *Computers and Electronics in Agriculture* 72.1, p. 48 -52. ISSN : 0168-1699. DOI : https://doi.org/10.1016/j.compag.2010.02.002. URL : http://www.sciencedirect.com/science/article/pii/S0168169910000384.

MUNKRES, James (1957). « Algorithms for the assignment and transportation problems ». In : *Journal of the society for industrial and applied mathematics* 5.1, p. 32-38.

MURTAGH, Fionn et Pedro CONTRERAS (2011). « Methods of hierarchical clustering ». In : *arXiv preprint arXiv :1105.0121*.

NÄÄS, Irenilza de A et al. (2012). « Image analysis for assessing broiler breeder behavior response to thermal environment ». In : *Engenharia Agrícola* 32.4, p. 624-632.

NELDER, John Ashworth et Robert WM WEDDERBURN (1972). « Generalized linear models ». In : *Journal of the Royal Statistical Society : Series A (General)* 135.3, p. 370-384.

NEVES, Diego Pereira et al. (2015). « Detection of flock movement and behaviour of broiler chickens at different feeders using image analysis ». In : *Information Processing in Agriculture* 2.3, p. 177 -182. ISSN : 2214-3173. DOI : https://doi.org/10.1016/j.inpa.2015.08.002. URL : http://www.sciencedirect.com/science/article/pii/S2214317315000475.

NIGSCH, Florian et al. (2006). « Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization ». In : *Journal of chemical information and modeling* 46.6, p. 2412-2422.

NIWATTANAKUL, Suphakit et al. (2013). « Using of Jaccard coefficient for keywords similarity ». In : *Proceedings of the international multiconference of engineers and computer scientists*. T. 1. 6, p. 380-384.

O'MAHONY, Niall et al. (2019). « Deep learning vs. traditional computer vision ». In : *Science and Information Conference*. Springer, p. 128-144.

PAN, Sinno Jialin et Qiang YANG (2009). « A survey on transfer learning ». In : *IEEE Transactions on knowledge and data engineering* 22.10, p. 1345-1359.

PARTIO, Mari et al. (2002). « Rock texture retrieval using gray level co-occurrence matrix ». In : *Proc. of 5th Nordic Signal Processing Symposium*. T. 75. Citeseer.

PAULINE, C et al. (2019). « Using the EBENE method to assess welfare and health of broilers. » In : *13èmes Journées de la Recherche Avicole et Palmipèdes à Foie Gras, Tours, France, 20 et 21 mars 2019*, p. 385-389.

PELE, Ofir et Michael WERMAN (2010). « The quadratic-chi histogram distance family ». In : *European conference on computer vision*. Springer, p. 749-762.

PEREIRA, Danilo F. et al. (2013). « Machine vision to identify broiler breeder behavior ». In : *Computers and Electronics in Agriculture* 99, p. 194 -199. ISSN : 0168-1699. DOI : https://doi.org/10.1016/j.compag.2013.09.012. URL : http://www.sciencedirect.com/science/article/pii/S0168169913002299.

PIRSIAVASH, Hamed, Deva RAMANAN et Charless C FOWLKES (2011). « Globally-optimal greedy algorithms for tracking a variable number of objects ». In : *CVPR 2011*. IEEE, p. 1201-1208.

RANGARAJAN, Krishnan et Mubarak SHAH (1991). « Establishing motion correspondence ». In : *CVGIP : image understanding* 54.1, p. 56-73.

REDMON, Joseph et al. (2016). « You only look once : Unified, real-time object detection ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 779-788.

REID, D. (1979). « An algorithm for tracking multiple targets ». In : *IEEE Transactions on Automatic Control* 24.6, p. 843-854. DOI : 10.1109/TAC.1979.1102177.

REID, Donald (1979). « An algorithm for tracking multiple targets ». In : *IEEE transactions on Automatic Control* 24.6, p. 843-854.

REN, Shaoqing et al. (2015). « Faster r-cnn : Towards real-time object detection with region proposal networks ». In : *Advances in neural information processing systems* 28, p. 91-99.

REZATOFIGHI, Seyed Hamid et al. (2015). « Joint probabilistic data association revisited ». In : *Proceedings of the IEEE international conference on computer vision*, p. 3047-3055.

RONNEBERGER, Olaf, Philipp FISCHER et Thomas BROX (2015). « U-net : Convolutional networks for biomedical image segmentation ». In : *International Conference on Medical image computing and computer-assisted intervention*. Springer, p. 234-241.

ROSENBLATT, Frank (1958). « The perceptron : a probabilistic model for information storage and organization in the brain. » In : *Psychological review* 65.6, p. 386.

ROSTEN, Edward et Tom DRUMMOND (2006). « Machine learning for high-speed corner detection ». In : *European conference on computer vision*. Springer, p. 430-443.

SALARI, V. et I. K. SETHI (1990). « Feature point correspondence in the presence of occlusion ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.1, p. 87-91. DOI : 10.1109/34.41387.

SALES, GT et al. (2015). « Quantifying detection performance of a passive low-frequency RFID system in an environmental preference chamber for laying hens ». In : *Computers and Electronics in Agriculture* 114, p. 261-268.

SAVAS, Berkant et Lars ELDÉN (2007). « Handwritten digit classification using higher order singular value decomposition ». In : *Pattern recognition* 40.3, p. 993-1003.

SCHWEITZER, Haim, James Wesley BELL et Feng WU (2002). « Very fast template matching ». In : *European Conference on Computer Vision*. Springer, p. 358-372.

SERGEANT, D., R. BOYLE et M. FORBES (1998). « Computer visual tracking of poultry ». In : *Computers and Electronics in Agriculture* 21.1, p. 1 -18. ISSN : 0168-1699. DOI : https://doi.org/10.1016/S0168-1699(98)00025-8. URL : http://www.sciencedirect.com/science/article/pii/S0168169998000258.

SETHI, Ishwar K et Ramesh JAIN (1987). « Finding trajectories of feature points in a monocular image sequence ». In : *IEEE Transactions on pattern analysis and machine intelligence* 1, p. 56-73.

SHARMA, Sagar, Simone SHARMA et Anidhya ATHAIYA (2017). « Activation functions in neural networks ». In : *towards data science* 6.12, p. 310-316.

SHI, Jianbo et al. (1994). « Good features to track ». In : *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, p. 593-600.

SIEGFORD, Janice M et al. (2016). « Assessing activity and location of individual laying hens in large groups using modern technology ». In : *Animals* 6.2, p. 10.

SINGH, Gurinderbeer, Sreeraman RAJAN et Shikharesh MAJUMDAR (2017). « A greedy data association technique for multiple object tracking ». In : *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*. IEEE, p. 177-184.

SONG, Bi et al. (2010). « A stochastic graph evolution framework for robust multi-target tracking ». In : *European Conference on Computer Vision*. Springer, p. 605-619.

SPONTÓN, Haldo et Juan CARDELINO (2015). « A review of classic edge detectors ». In : *Image Processing On Line* 5, p. 90-123.

SPRINGENBERG, Jost Tobias et al. (2014). « Striving for simplicity : The all convolutional net ». In : *arXiv preprint arXiv :1412.6806*.

SUTHAHARAN, Shan (2016). « Support vector machine ». In : *Machine learning models and algorithms for big data classification*. Springer, p. 207-235.

SZEGEDY, Christian et al. (2016). « Rethinking the inception architecture for computer vision ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2818-2826.

SZEGEDY, Christian et al. (2017). « Inception-v4, inception-resnet and the impact of residual connections on learning ». In : *Thirty-first AAAI conference on artificial intelligence*.

TUCERYAN, Mihran et Anil K JAIN (1993). « Texture analysis ». In : *Handbook of pattern recognition and computer vision*, p. 235-276.

TURETSKY, R (1951). « The least squares solution for a set of complex linear equations ». In : *Quarterly of Applied Mathematics* 9.1, p. 108-110.

VADDI, RS et al. (2011). « Contour detection using freeman chain code and approximation methods for the real time object detection ». In : *Asian Journal of Computer Science and Information Technology* 1.1, p. 15-17.

VAN HERTEM, Tom et al. (2018). « Predicting broiler gait scores from activity monitoring and flock data ». In : *Biosystems Engineering* 173. Advances in the Engineering of Sensor-based Monitoring and Management Systems for Precision Livestock Farming, p. 93 -102. ISSN : 1537-5110. DOI : https://doi.org/10.1016/j.biosystemseng.2018.07.002. URL : http://www.sciencedirect.com/science/article/pii/S1537511018304616.

VEENMAN, Cor J, Marcel JT REINDERS et Eric BACKER (2001). « Resolving motion correspondence for densely moving points ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.1, p. 54-72.

WANG, Qiang et al. (2019). « Fast online object tracking and segmentation : A unifying approach ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 1328-1338.

WARD JR, Joe H (1963). « Hierarchical grouping to optimize an objective function ». In : *Journal of the American statistical association* 58.301, p. 236-244.

WEI, Yichen et al. (2007). « Interactive offline tracking for color objects ». In : *2007 IEEE 11th International Conference on Computer Vision*. IEEE, p. 1-8.

WENG, Shiuh-Ku, Chung-Ming KUO et Shu-Kang TU (2006). « Video object tracking using adaptive Kalman filter ». In : *Journal of Visual Communication and Image Representation* 17.6, p. 1190-1208.

WOLD, Svante, Kim ESBENSEN et Paul GELADI (1987). « Principal component analysis ». In : *Chemometrics and intelligent laboratory systems* 2.1-3, p. 37-52.

YAN, Fei et al. (2006). « A novel data association algorithm for object tracking in clutter with application to tennis video analysis ». In : *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. T. 1. IEEE, p. 634-641.

YANG, Ming, Ting YU et Ying WU (2007). « Game-theoretic multiple target tracking ». In : *2007 IEEE 11th International Conference on Computer Vision*. IEEE, p. 1-8.

YILMAZ, Alper, Omar JAVED et Mubarak SHAH (2006). « Object tracking : A survey ». In : *Acm computing surveys (CSUR)* 38.4, 13-es.

YOON, Kwangjin et al. (2019). « Data association for multi-object tracking via deep neural networks ». In : *Sensors* 19.3, p. 559.

ZHANG, Li, Yuan LI et Ramakant NEVATIA (2008). « Global data association for multi-object tracking using network flows ». In : *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, p. 1-8.

ZHANG, Lu et Laurens VAN DER MAATEN (2013). « Preserving structure in model-free tracking ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.4, p. 756-769.

ZHAO, Zhong-Qiu et al. (2019). « Object detection with deep learning : A review ». In : *IEEE transactions on neural networks and learning systems* 30.11, p. 3212-3232.

ZHOU, Feng, Ju Fu FENG et Qing Yun SHI (2001). « Texture feature based on local Fourier transform ». In : *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*. T. 2. IEEE, p. 610-613.

ZHUANG, Xiaolin et al. (2018). « Development of an early warning algorithm to detect sick broilers ». In : *Computers and Electronics in Agriculture* 144, p. 102-113.