Détection d'animaux en forêt par vision artificielle

A. Gerardeaux

Clermont-Ferrand

U.F.R. Sciences et Technologies

MÉMOIRE DE STAGE

1ère année de Master

Spécialité : Automatique-Robotique

DÉTECTION D'ANIMAUX EN FORET PAR VISION ARTIFICIELLE

Stagiaire : **Antoine GERARDEAUX**

Tuteur : **Bernard BENET**

Centre Irstea de Clermont-Ferrand-Site d'Aubière

Du 19 avril 2018 au 19 juillet 2018

Remerciement

Je tiens à remercier l'ensemble des personnels d'Irstea pour l'accueil au sein du centre de Clermont-Ferrand.

Je remercie tout particulièrement mon responsable Irstea M. Bernard BENET pour son encadrement, l'aide et les conseils qu'il a pu me fournir tout au long de ce projet, ainsi que M. Yves BOSCARDIN et M. Anders MÅRELL qui ont participé à mon encadrement et enfin M. Vincent ROUSSEAU pour son aide et ses conseils techniques sur le plan informatique.

Enfin, j'adresse mes remerciements à mon tuteur UCA, M. Thierry CHATEAU pour son aide et son suivi ainsi qu'à M. Pierre-Yves LACROIX.

Résumé de stage

Le cadre de mon sujet s'inscrit dans le travail de suivi du dispositif expérimental OPTMix qui est un groupe de chercheurs IRSTEA basé à Nogent-sur-Vernisson. OPTMix vise à améliorer les connaissances sur le fonctionnement des forêts mélangées en région tempérée avec des applications directes à la gestion forestière en particulier dans le cadre des changements climatiques. Trois facteurs sont contrôlés :

- La composition du peuplement ;
- La densité du peuplement ;
- La présence des herbivores (espèces, heures de passage, nombre d'individus, type d'activités). C'est sur ce dernier point que je suis amené à intervenir dans le cadre de mon stage.

Le but de mon stage est l'amélioration, l'optimisation, la modification de programmes C++ et python pour la détection d'animaux par vision artificielle en milieu forestier avec l'utilisation et la création de fonctions (OpenCV). De plus, des comparaisons de résultats de détection manuelle (fastidieuses et contraignantes) avec des résultats obtenus à l'aide l'algorithme de traitement d'image sont effectuées afin de mettre en valeur l'intérêt d'une méthode automatique. L'objectif est de développer un algorithme permettant de détecter tous les animaux sur les images, en conservant un niveau de bruit acceptable et une robustesse vis à vis des différents environnements rencontrés (parcelle de forêt sombre en février, ou très verdoyante en juillet, résineux, pur chêne, mixte avec des pins) ainsi que les différentes espèces présentes dans ces secteurs (sanglier, cerf, chevreuil, pigeon, rapace, renard).

Sommaire

I. Introduction

Dans le cadre du stage de 1ère année du Master Automatique et Robotique, parcours perception artificielle et robotique de l'université Clermont Auvergne, j'ai été accueilli par le centre de recherche Irstea de Clermont-Ferrand, basée à Aubière, pour une durée de quatre mois. L'Irstea est un institut de recherche public à caractère scientifique et technologique dans les domaines de l'environnement et de l'agriculture.

L'unité de recherche Technologies et Systèmes d'information pour les agrosystèmes (TSCF) d'Irstea a pour objectif de répondre aux besoins d'une agriculture productive et responsable par le biais du développement de solutions robotiques et informatiques visant à apporter une aide à l'agriculteur dans son travail. Au sein de cette unité de recherche, l'équipe RObotique et Mobilité pour l'Environnement et l'Agriculture (ROMEA) dont j'ai fait partie durant ce stage, conçoit des systèmes adaptés, pour accroître les performances et la sécurité des engins œuvrant dans les milieux agricoles.

Dans ce contexte, Bernard Benet ingénieur/chercheur en traitement d'image et perception m'a donné l'opportunité d'effectuer mon stage dans les locaux d'Aubière au sein de son équipe. Les différents projets pour lesquels un membre de l'équipe ROMEA peut être engagé sont nombreux et ne sont pas tous liés aux problématiques de l'agriculture. En effet le dispositif expérimental de suivi sur lequel je suis amené à intervenir se nomme OPTMix (Oak Pine Tree Mixture) et est piloté par Nathalie Korboulewsky. Basé à Nogent-sur-Vernisson, à quelques kilomètres au sud de Paris en forêt domaniale d'Orléans, l'équipe FONA, plus particulièrement Yves Boscardin, étudie le suivi de la pression par les herbivores, et plus précisément dans le cadre du projet OPTMix, plusieurs dispositifs expérimentaux sont disposés afin de contrôler trois facteurs principaux : la composition du peuplement, la densité du peuplement et la présence des herbivores (espèces, heures de passage, nombres d'individus, type activités), et c'est sur ce dernier point que je suis amené à intervenir.

L'étude de la présence des herbivores sur les placettes expérimentales forestières permet de mieux comprendre, d'analyser et de quantifier l'influence de ce facteur sur la biodiversité et donc sur le renouvellement des peuplements. C'est pourquoi depuis plus de 15 ans des pièges photographiques sont disposés sur le terrain afin de recenser le passage des animaux. Une fois les données récupérées sur un format vidéo, les agents de Nogent sont amenés à faire un travail de traitement manuel des vidéos afin d'extraire manuellement les informations pour leurs expérimentations. C'est un travail fastidieux et répétitif.

La solution proposée par mon maître de stage est un algorithme de traitement d'image, robuste, rapide, utilisable instinctivement et capable d'extraire uniquement les informations nécessaires pour alimenter en données le dispositif. Ainsi, après avoir exposé plus en détail le contexte, je présenterai d'abord la partie traitement d'image et le travail réalisé d'optimisation de l'existant. Ensuite j'aborderai l'implémentation du Deep Learning qui permet d'améliorer considérablement les résultats

II. Présentation de l'entreprise et du service d'accueil

1. Présentation de l'entreprise

A. Carte d'identité

- IRSTEA : Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture
- Adresse : 9, avenue Blaise Pascal CS 20085 63178 Aubière
- Téléphone : +33 (0)4 73 44 06 00
- Date de création : 2012
- SIRET : 180 070 013 00198
- Statut juridique : Etablissement public national à caractère administratif
- Budget 2016 : 109,5 millions d'euros
- Effectif total en 2016 : 1129 chercheurs, ingénieurs, doctorants et post-doctorants ainsi que 1533 collaborateurs statutaires et contractuels.

***Figure n°1** : Les centres, leurs unités de recherche et les partenariats structurants en région*

B. Historique

En 1971, les centres nationaux d'études techniques et de recherches technologiques pour l'agriculture, les forêts et l'équipement rural (CERAFER) furent créés afin de mener des études dans différents domaines, notamment l'agriculture de montagne, le suivi des innovations techniques, ou les problèmes liés à l'utilisation ou à la maîtrise de l'eau. Dès 1973, la dénomination change et devient Centre technique du génie rural des eaux et des forêts (CTGREF).

L'appellation de la structure change et devient EPA en 1982 avec la dénomination Centre national du machinisme agricole du génie rural, des eaux et des forêts (CEMAGREF). Elle se transforme en établissement public à caractère scientifique et technologique (EPST) en 1986 sous la double tutelle des ministères chargés de l'agriculture et de la recherche. C'est en février 2012 que la structure est nommée Irstea, Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture.

C. Implantation géographique

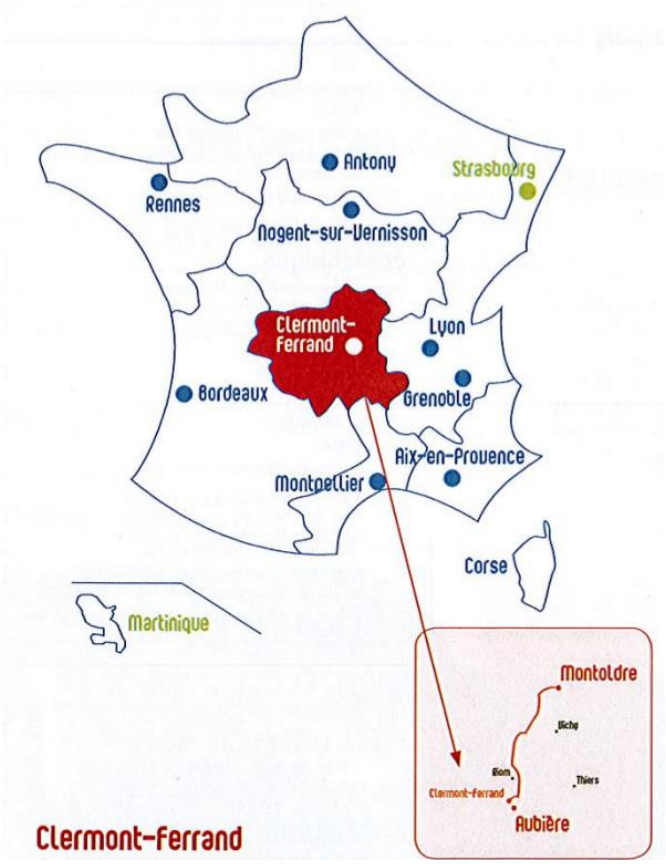


Figure n°2 *: Carte de l'implantation géographique*

D. Activités

Le plan stratégique d'Irstea vise à répondre aux principaux enjeux socio-économiques en lien avec :

- La gestion de la durabilité des territoires, notamment agricoles et périurbains, leurs ressources (en particulier l'eau), leurs productions (alimentaire et énergétique), mais aussi les flux humains, économiques et financiers ;
- La prévision et la prévention des risques naturels (crues, inondations, avalanches) et environnementaux (chimiques, biologiques) ;
- La préservation de la biodiversité et sa participation à la production de ressources (biomasse forestière).

Pour cela, l'institut concentre ses recherches autour de 12 thèmes mis en œuvre dans un ou plusieurs centres **[1]**. Pour le département Eaux, les thèmes de recherche sont :

- Aléas et risques liés au cycle de l'eau (ARCEAU) ;
- Réponses biologiques et écologiques aux contaminants du milieu aquatique (BELCA) ;
- Gestion de l'eau, des usages, des services et des infrastructures (GEUSI) ;
- Qualité des systèmes aquatiques et restauration écologique (QUASARE) ;
- Risques liés aux phénomènes gravitaires rapides et sûreté des ouvrages hydrauliques et de protection (RIVAGE).

Pour le département Territoires, nous avons :

- Développement territorial et agriculture multifonctionnelle (DTAM) Systèmes d'information spatiale pour la gestion intégrée de l'environnement (SYNERGIE) ;
- Systèmes écologiques terrestres : dynamique, vulnérabilités et ingénierie (SEDYVIN).

Et pour le département Écotechnologies :

- Innovation technologique au service de l'agriculture et de l'environnement (INSPIRE) ;
- Modèles, systèmes d'information et gestion viable de l'environnement (MOTIVE) Structures, procédés, écoulement, énergie (SPEE) ;
- Technologies et procédés pour l'eau et les déchets (TED).

Irstea regroupe plus de 1500 collaborateurs dont 1100 chercheurs, ingénieurs, Doctorants et post-doctorants répartis sur 9 centres de recherche dans toute la France : Aix-en-Provence (13), Antony (92), Bordeaux (33), Clermont-Ferrand (63), Grenoble (38), Lyon (69), Montpellier (34), Nogent-sur-Vernisson (45) et Rennes (35).

E. Structure interne

Le Centre de Clermont-Ferrand, se compose de 90 agents permanents (dont 60 ingénieurs/chercheurs), et accueille chaque année environ 22 doctorants et post-doctorants et 40 stagiaires de l'enseignement supérieur. Il bénéficie de 3,56 millions d'euros de budget annuel en moyenne.

Le centre dispose de deux sites : un pôle scientifique et universitaire situé sur le campus des Cézeaux à Aubière (63), et un site de recherche et d'expérimentation (équipé entre autres d'une exploitation agricole et d'un atelier de prototypage mécanique) situé à Montoldre (03). Dans le cadre de mon stage, j'ai été rattaché à l'équipe ROMEA de l'unité de recherche TSCF que je vais vous présenter brièvement afin de pouvoir procéder à la présentation de l'équipe Fona et du projet OPTMix. Vous trouverez ci-dessous l'organigramme du Centre de Clermont-Ferrand **Fig. 3. Organigramme du Centre de Clermont-Ferrand**

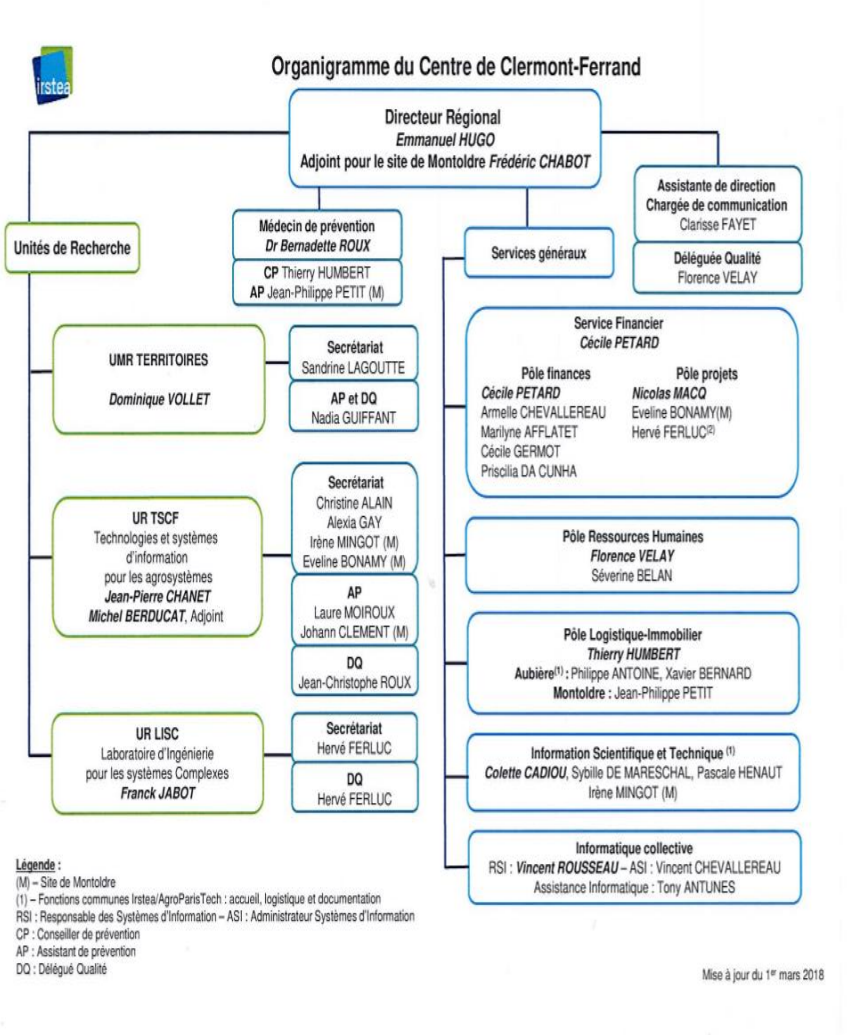


***Figure n°3** : Organigramme du Centre de Clermont-Ferrand*

2. Présentation du service accueillant

A. Unité de recherche TSCF

L'unité de recherche (UR) Technologies et Systèmes d'information pour les agrosystèmes (TSCF) **[2]** fait partie de l'une des 19 unités de recherches (dont 5 unités mixtes de recherches) que compte actuellement Irstea. Ses 70 agents titulaires sont répartis dans 4 équipes de recherche, et s'intéressent au domaine de la sécurité et des performances des agroéquipements pour apporter des réponses concrètes aux besoins d'une agriculture productive écologiquement responsable. Ses activités qui relèvent du département Écotechnologies d'Irstea sont conduites dans le cadre des thèmes de recherche « INSPIRE » et « MOTIVE ».

B. Équipe ROMEA

Au sein de l'unité de recherche TSCF, l'équipe « RObotique et Mobilité pour l'Environnement et l'Agriculture » (ROMEA) **[3]** conçoit des systèmes reconfigurables et à autonomie partagée, afin d'accroître les performances et la sécurité des engins œuvrant en milieux naturels, en particulier dans le milieu de l'agriculture. Dirigée par M. Roland LENAIN, cette équipe concentre ses recherches sur trois principaux axes :

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

C. Équipe Fona

Bien que réalisant mon stage au sein de l'équipe ROMEA dont fait partie mon maître de stage Bernard Benet, le cadre comme le contrat de travail sont fournis par l'équipe Fona d'Irstea **[4]** basée à Nogent-sur-Vernisson à 115 km au sud de Paris, en pleine forêt **Fig. 4. Implantation géographique de l'équipe Fona.**

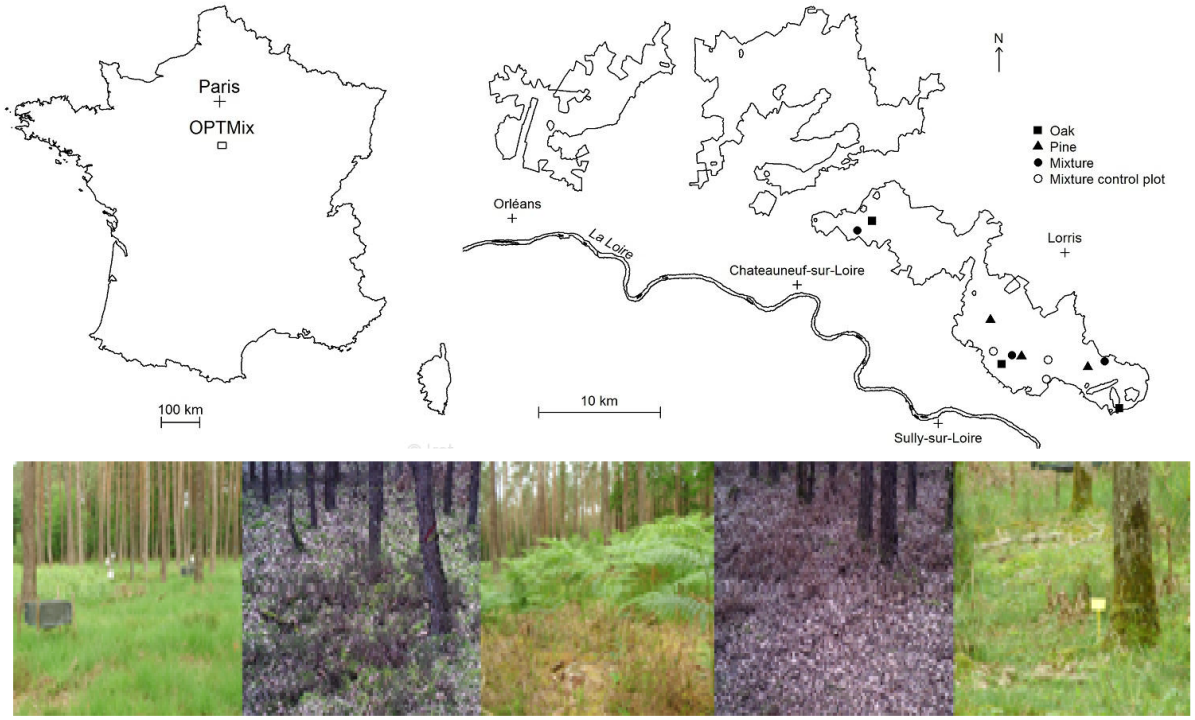


Figure n°4 *: Implantation géographique de l'équipe Fona*

L'équipe Fona est composée de 8 acteurs, dont Anders Mårell, chercheur en écologie végétale et animateur de l'équipe ainsi que Yves Boscardin, Assistant ingénieur spécialiste de la dynamique forestière avec lesquels j'ai été amené à collaborer.

L'ambition de cette équipe se décline principalement sous ces 4 points :

- Comment adapter le renouvellement de la forêt en présence d'ongulés sauvages dans un contexte de changement climatique ?
- Quels sont les effets des ongulés sauvages sur le fonctionnement et la biodiversité des écosystèmes forestiers ?
- Quels sont les mécanismes par lesquels les ongulés sauvages modifient la dynamique des écosystèmes ?
- Quels sont les outils d'aide à la gestion proposée pour mieux gérer les populations d'ongulés sauvages et leurs effets sur le milieu ?

D. Le projet OPTMix

Le dispositif expérimental OPTMix **[5]** regroupe plusieurs parcelles expérimentales, formant un réseau de 40 hectares de placettes forestières visant à étudier l'influence de la densité, et de la composition du peuplement ainsi que du passage des herbivores sur la biodiversité. Ce dispositif vise donc à améliorer les connaissances sur le fonctionnement des forêts mélangées en région tempérée avec des applications directes à la gestion forestière particulièrement dans le cadre des changements climatiques.

Ce dispositif est donc constitué d'enclos et d'exclos afin de pouvoir comparer une même végétation à différentes fréquences de passage des animaux, en empêchant les grands de rentrer dans les enclos grâce à des grillages. Les appareils photographiques permettant le suivi des placettes seront donc ma principale source de données pour alimenter mon algorithme. La plupart des placettes sont aussi équipées d'appareils de mesure du microclimat, en particulier du bilan hydrique (capteurs de température, d'humidité relative, de rayonnement, pluviomètres, sondes d'humidité du sol à trois profondeurs, piézomètres, dendromètres). Le schéma ci-dessous résume la chaine de traitement des données de pièges photographiques **Fig. 5. Schéma d'organisation du projet OPTMix**

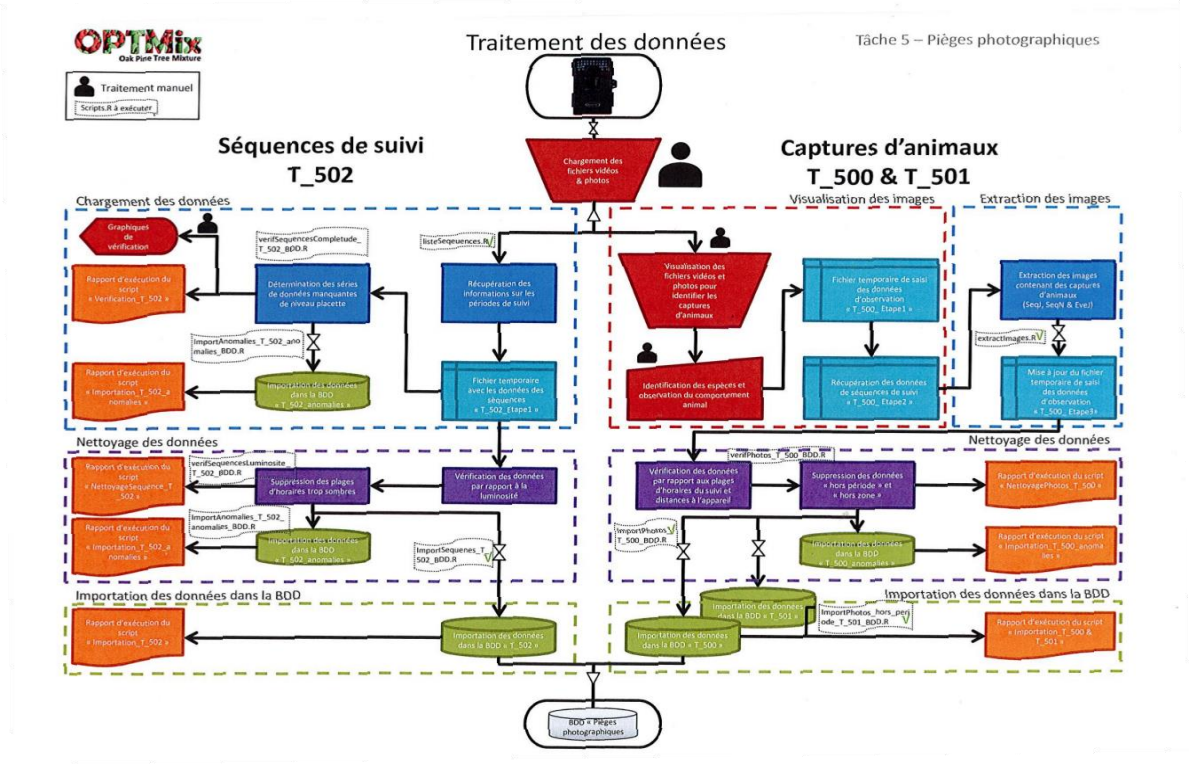


***Figure n°5** : Schéma d'organisation du projet OPTMix*

III. Présentation du poste et de l'étude confiée

1. Présentation du poste

A. Tâches

L'objectif de ce stage est donc la réalisation d'un algorithme de traitement d'image robuste, rapide, et utilisable instinctivement capable de détecter tout passage d'animaux en conservant un niveau de bruit acceptable face à une scène variable dans le temps et dans l'espace.

Des techniques basiques de traitements d'image se sont montrées inefficaces dû à la complexité du milieu (fortes variations de lumière, milieu encombré de végétation variable, irrégularités des scènes de calculs, irrégularité des couleurs dominantes etc.). Le but de mon stage est donc l'optimisation et la modification de l'algorithme existant par l'ajout ou la création de nouvelles fonctions, l'implémentation de solutions complémentaires et la comparaison avec des résultats manuels. L'objectif est donc de ne rater aucun animal tout en gardant un niveau de bruit acceptable voir nul, dans l'idéal. Les langages utilisés seront le C++ ainsi que le Python en utilisant principalement la bibliothèque OpenCV **[6]** de traitement d'image, et Tensorflow pour le Deep Learning. A l'issue de mon stage, nous fournirons aux agents de l'équipe FONA une machine prête à l'utilisation. Cela permettra directement de bénéficier des fonctionnalités de l'algorithme et donc de l'utiliser.

Nous avons été amenés à nous déplacer. D'abord à différent séminaires, notamment sur le Deep Learning, mais aussi sur le terrain afin d'affiner notre solution aux besoins spécifiques du suivi. C'est donc dans la forêt domaniale d'Orléans, au centre de recherche d'Irstea à Nogent-sur-Vernisson, que Bernard Benet et moi-même, nous nous sommes rendus. Nous avons rencontré les acteurs du projet OPTMix et échangé sur différents points importants comme la présentation du dispositif et des placettes expérimentales et l'élaboration de notre travail.

B. Environnement humain

Dans le cadre de ce stage j'ai été encadré par Bernard Benet, ingénieur/chercheur, spécialiste en traitement d'image et perception ainsi que Thierry Château, enseignant chercheur à l'Université Clermont Auvergne.

Bernard Benet *Thierry Chateau*

Les personnes avec lesquelles j'ai été amené à travailler appartenant à l'équipe FONA sont Anders Mårell, chercheur en écologie végétale et Yves Boscardin, assistant ingénieur spécialiste de la dynamique forestière.

Anders Mårell *Yves Boscardin*

J'ai aussi eu l'occasion de bénéficier de l'aide de Vincent Rousseau, informaticien responsable de la forge et spécialiste du développement ROS (Robot Operating System). Enfin j'ai eu le soutien de Pierre-Yves Lacroix, ingénieur à l'institut Pascal de Clermont-Ferrand.

C. Environnement matériel

Quasiment la totalité du travail demandé sera orienté software. J'ai donc eu à disposition un ordinateur portable professionnel sous un environnement Linux (Ubuntu). Dans un premier temps, le langage utilisé était le C++ compilable avec ROS. Ensuite le programme marchera en python et la majorité des fonctions importantes seront utilisées avec la bibliothèque OpenCV.

Le flux de données reçu qui nous permettra d'implémenter et d'optimiser la solution provient de pièges photographiques posés stratégiquement afin de couvrir intégralement une placette. Ce flux de données est reçu sous forme d'un format vidéo en .mp4 ou encore en .AVI de environs 240 images prises successivement toutes les minutes, avec une autonomie de 100 heures. Les caméras sont aussi équipées d'un système de détection offrant la possibilité de prendre une image entre deux minutes successives lorsqu'un mouvement a été détecté **Fig. 6. Appareil de suivi photographique. [7]**

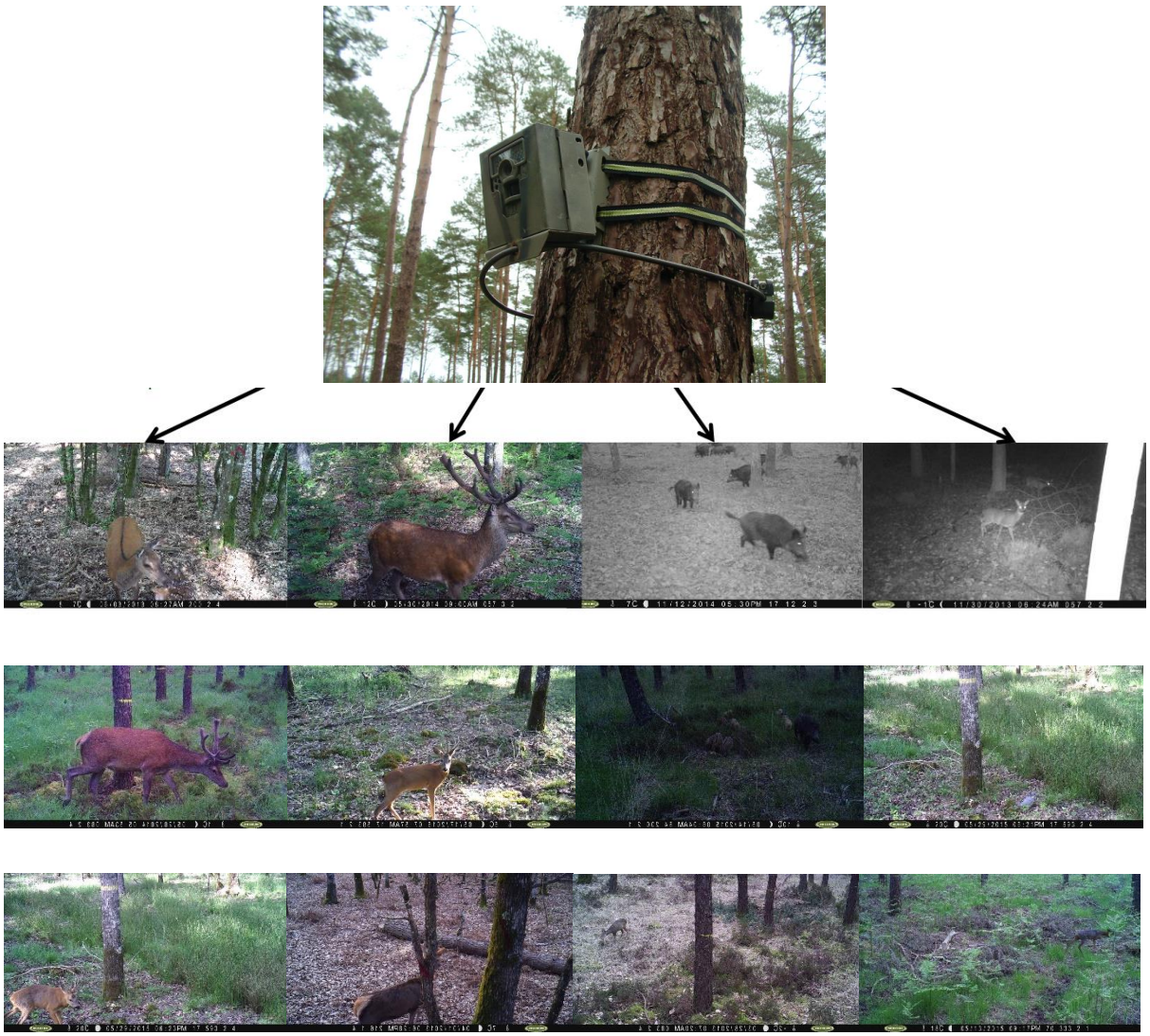


***Figure n°6** : Appareil et suivi photographique*

2. Présentation de l'étude

A. Réalisation par traitement d'image

a. Splitting des données

Comme expliqué précédemment, l'appareil qui permet aux agents de Nogent-sur-Vernisson de récupérer leurs bases de données fournit un format vidéo comportant minimum 240 images. Pour une parcelle donnée, la caméra est posée successivement à chaque coin de la passerelle, pour une durée totale de deux semaines de surveillance par parcelles, effectuée plusieurs fois par an sur une trentaine de parcelles différentes. Le flux de donnée est donc très vaste et nécessite de longues heures d'observation chaque année et ce, depuis de nombreuses années. Afin de pouvoir traiter toutes les vidéos à la suite pour une année par exemple, sans avoir à relancer le programme à chaque fois, il a été nécessaire, au préalable, de transformer les vidéos en images successives (méthode de Splitting) **Fig. 4. Splitting en images.**

J'ai donc dans un premier temps implémenter en amont du futur traitement, la possibilité de traiter successivement les images coupées présentes dans un dossier tampon, ce qui permet de stocker les 240 images en cours de traitement issues des fichiers vidéo, puis de passer automatiquement à la vidéo d'après en écrasant les 240 images par les 240 nouvelles, ce qui permet de réduire l'enregistrement mémoire.

Le traitement serait donc effectué à chaque fois sur les images successives de nos vidéos. Je viendrais donc interroger les images, une par une, et détecter, ainsi que compter, la présence ou non d'un ou plusieurs animaux sur la scène Pour cela j'ai eu à créer une fonction exécutée à chaque début de programme avec une indentation permettant de traiter successivement toutes les vidéos. Cette fonction est très simple et utilise essentiellement des opérations classiques de gestion de fichiers photos et vidéos. Vous trouverez en annexe 1 le code de cette fonction élémentaire de notre algorithme.

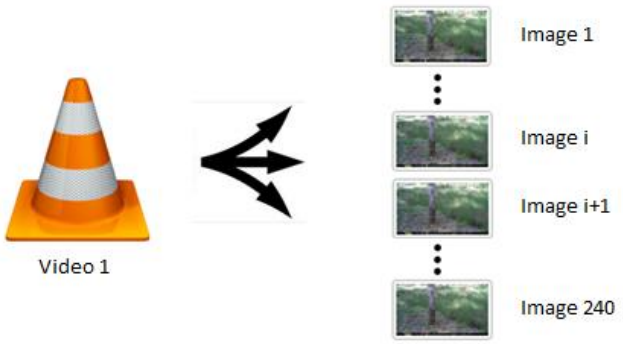


***Figure n°7** : Splitting en images*

b. Utilisation d'OpenCV

Dans un premier temps des techniques classiques de traitement d'image **[8]** sont utilisées sur les images successives. Une différence de niveau de gris va nous permettre de mettre en évidence, d'une image à l'autre, l'apparition ou la disparition d'un animal. L'image i et l'image i+1 sont chargées dans l'algorithme à l'aide de fonctions de bases puis transformées en niveaux de gris grâce à une fonction très classique intitulé BGR_TO_GRAY. Comme son nom l'indique elle permet de passer d'une image couleur en base (RGB) à une image en nuance de gris. Puis nous effectuons la différence entre les valeurs de chaque pixel. Le résultat de cette différence de gris est à la base même de notre programme, et donc de notre première opération **Fig. 8. Différence de niveaux de gris**. Cela permet d'extraire de l'information jugée intéressante à la résolution de notre problème. Le code est en annexe 2.

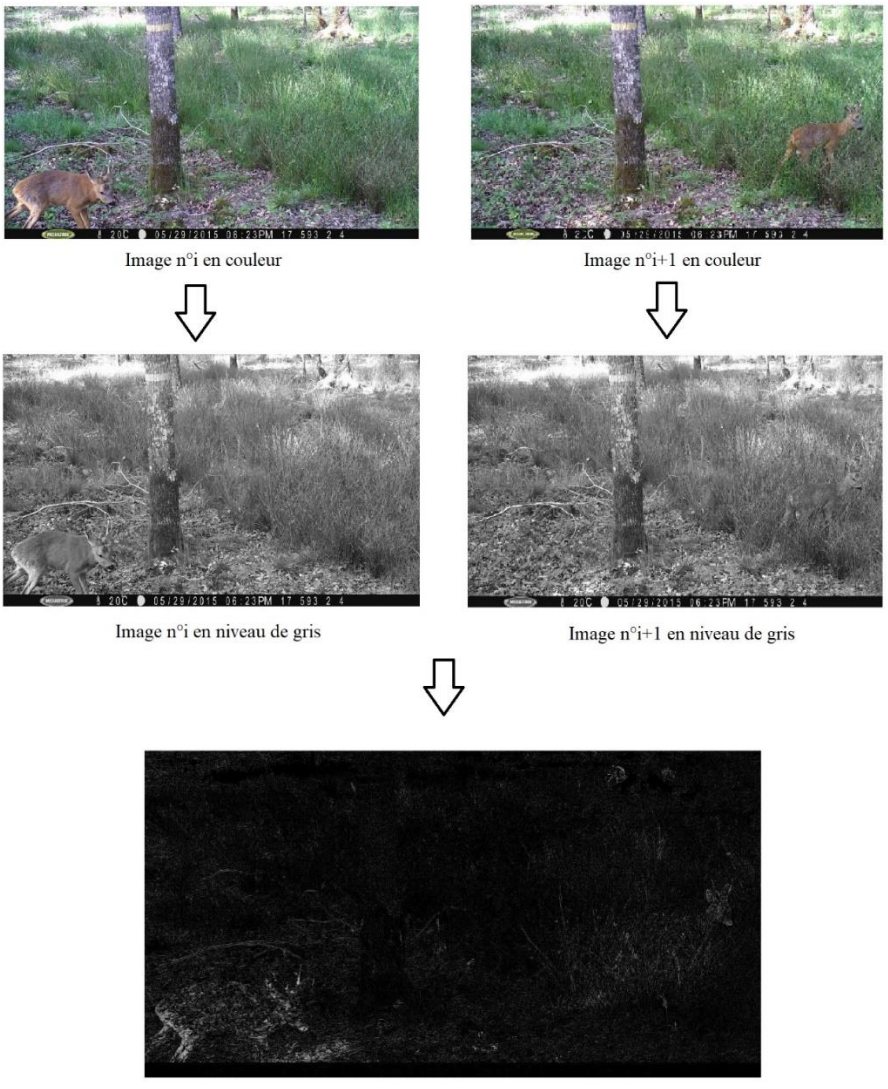


***Figure n°8** : Différence de niveaux gris*

Bien évidemment l'utilisation unique de cette différence est très loin d'être suffisante car le milieu forestier est un milieu très dynamique dans lequel, d'une minute à l'autre, beaucoup d'éléments peuvent avoir bougés sans que ce soit, pour autant, de la vie animale (changement de luminosité, mouvement de la végétation etc.). C'est pour cela que, par la suite, une série d'opération sur l'image va permettre, via plusieurs seuillages, de ne faire ressortir que le mouvement de la faune, éliminant les bruits, d'une manière ou d'une autre, en se basant sur la nature de leur provenance.

Afin d'éliminer une grande partie des bruits identifiés, nous faisons une binarisation de l'image. Elle permet d'obtenir des valeurs de pixels ne valant que 0 ou 1, c'est à dire blanc ou noir. Un seuillage est effectué afin de définir à partir de quelle valeur un pixel est blanc ou noir **Fig. 9. Effet de la binarisation et du seuillage**. Ce seuillage est indispensable et est probablement le plus important. Sans lui, il est impossible de discerner des animaux avec d'autres mouvements de la scène. Ce seuillage ou 'threshold' est effectué à l'aide de la fonction correspondante d'OpenCV. Vous trouverez en annexe 2 la partie du code en question.

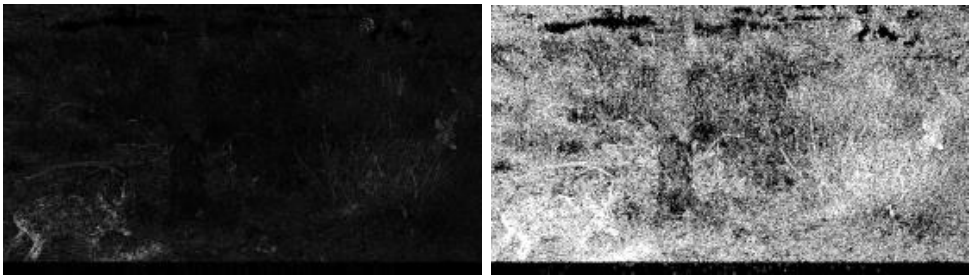


***Figure n°9** : Effet de la binarisation et du seuillage*

Ensuite, avec ce même objectif, à savoir réduire le bruit et affiner la détection, nous avons réalisé un seuillage des couleurs. Cela va permettre d'exclure des résultats trouvés, toutes les parties de l'image où les caractéristiques RGB d'un animal ne correspondent pas. Plus précisément, nous avons remarqué que les valeurs RGB des animaux que nous cherchons avaient, dans tous les cas, des valeurs de rouge plus élevées pour les mammifères que les valeurs de bleu et de vert, mais aussi des valeurs de bleu plus élevées que le rouge et le vert pour les rapaces et autres volatiles. Nous avons donc réglé le seuillage afin d'exclure les points ne respectant pas les critères de couleurs. Pour ce faire nous décomposons notre image en trois images correspondantes uniquement aux couleurs respectives RGB, nous pouvons donc opérer par soustraction de deux plans et ne faire ressortir que certaines caractéristiques de couleurs de l'image. Ainsi nous ne pouvions plus confondre le sol et d'autres éléments de la scène avec les animaux **Fig. 10. Effet du seuillage des couleurs.** Vous pourrez également trouver le code en annexe 3 et 4 ainsi que le détail en image du calcul effectué.

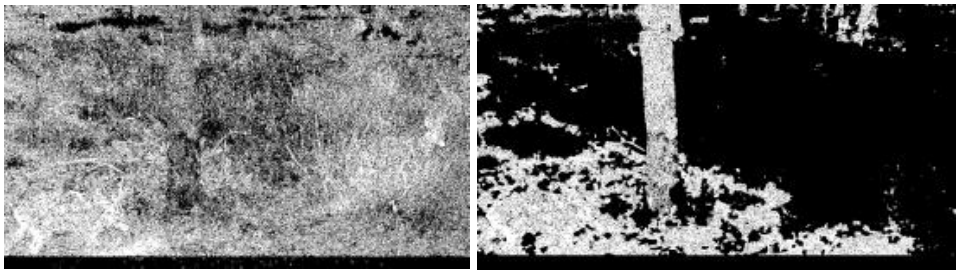


***Figure n°10** : Effet du seuillage des couleurs*

Enfin, nous réalisons une série d'opérations de morphologie d'image (filtres non-linéaires) afin d'affiner encore plus notre détection et par conséquent, réduire notre niveau de bruit. Dans un premier lieu nous procédons à une érosion puis une dilatation de l'image **Fig. 11. Effet de l'érosion puis de la dilatation puis du seuillage des labels**. Cette opération est très classique en traitement d'image et permet via l'érosion de faire disparaître les pixels isolés, puis via la dilatation de faire ressortir en opposition avec l'érosion les paquets les plus volumineux en matière de nombre de pixels. Juste à la suite nous caractérisons les paquets trouvés comme étant des labels indépendants et nous traitons les labels afin de faire disparaître les labels aberrants. Par exemple il serait considéré comme aberrant un label trop gros situé au dernier plan qui serait en opposition avec un label trop petit en fond de scène. Comme pour chaque étape de traitement, le code sera répertorié en annexe 3.

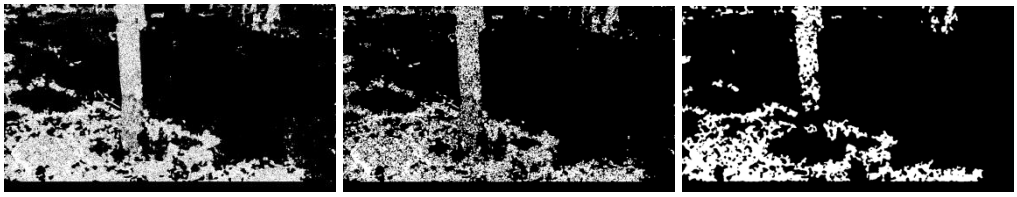


***Figure n°11** : Effet de l'érosion puis de la dilatation puis du seuillage des labels*

Nous avons aussi remarqué que l'application d'un filtre médian **Fig. 12. Effet du filtre médian,** avait pour effet d'augmenter la performance de détection. Ce filtre permet de moyenner et donc de traiter en plus certains points qui auraient été hors calcul. C'est un lissage des résultats très utile qui nous a permis d'avoir un paramètre sur lequel jouer en plus pour l'optimisation de notre algorithme.

***Figure n°12** : Effet du filtre médian*

c. Optimisation

Par la suite, nous avons réalisé un travail d'optimisation de l'algorithme. Cette phase est très importante, elle concerne l'amélioration du traitement des labels correspondants aux paquets détectés, via l'élimination de certains bruits caractéristiques ainsi que la classification supervisée via un algorithme annexe. Classification qui permet donc de discréditer certaines caractéristiques de l'image par rapport à d'autre que nous choisissons (sol/animaux ou arbres/animaux).

Dans un premier temps, partant du constat que les arbres sont une grande source de bruit qui est induite par la couleur sombre, et la teinte marron, proche de la couleur animale, j'ai eu à réaliser un algorithme en python permettant pour une scène donnée de sélectionner au préalable la position des arbres via des polygones, et stocker les valeurs dans un fichier texte qui lui est utilisé par le programme principal pour éliminer du traitement les zones en question **Fig. 13. Elimination polygonale des arbres.** Vous trouverez en annexes 4 et 5 le script du programme ainsi que son utilisation dans le programme principal. Cette fonction est contraignante car elle nécessite, en amont, un traitement. En revanche sur une utilisation du logiciel précise où la scène serait constante pour un grand nombre de vidéos, cette fonction supplémentaire pourrait s'avérer extrêmement utile.

***Figure n°13** : Elimination polygonale des arbres*

Ensuite un traitement reposant sur les caractéristiques des labels détectés est effectué. Il est réalisé, dans un premier temps, grâce à un calcul d'élongation. Ce dernier est fait par une fonction indépendante et permet de rejeter du traitement, des labels ayant une élongation supérieure à un seuil fixé pour filtrer les cas aberrants (longueur multipliée par largeur). Typiquement, nous déterminons le rectangle inscrit au label et nous récupérons les informations de position et de dimension afin de les traiter une à une **Fig. 14. Etude de l'élongation des paquets**. Vous trouverez la fonction en question et son utilisation dans le programme principal en annexe 6.

Avant utilisation de la fonction	Après l'utilisation de la fonction
	
	
	

Figure n°14 *: Etude de l'élongation des paquets*

Aussi, il a été mis en place, dans l'algorithme, une méthode permettant d'exclure les labels en se basant sur la caractéristique de surface. Nous excluons donc du traitement certains paquets en tenant compte du positionnement des labels par rapport au plan (premier plan, plan intermédiaire, dernier plan), les paquets aberrants qui, pour le premier plan, seraient trop petits, ou inversement trop grands pour une apparition en arrière-plan. Nous avons donc défini trois seuils différents en fonction du plan **Fig. 15. Décomposition en plans**. Le code en question est en annexe 7.

Figure n°15 *: Décomposition en plans*

De plus nous avons ajouté au programme une fonction supplémentaire permettant au traitement d'exclure et de comptabiliser les images trop sombres. Cette partie permet de se concentrer sur du temps de travail effectif et d'éviter de traiter des images inutiles. Nous avons pour cela utilisé une fonction qui calcule le niveau de gris moyen de l'image et donc permet de déterminer en valeur numérique, la quantité de lumière.

Afin d'optimiser la détection nous avons été amenés à utiliser un programme à part, permettant de générer un fichier de classification (méthode SVM) visant à la discrétisation de deux éléments d'une image appelés positif et négatif. Dans un premier temps nous avons testé plusieurs configurations de positifs/négatifs et tenter des couplages comme par exemple (animaux/arbre et animaux/sol). Bien que fastidieuse, cette phase de recherche fut, malheureusement, peu concluante ; cette technique semble efficace dans le cas d'une scène invariante en couleurs et en d'autres caractéristiques qui, pour notre cas, sont défavorables. Le milieu est très complexe car les couleurs du pelage ou bien du sol varient en fonction des saisons, des peuplements, des espèces et bien d'autres facteurs encore. Ayant été uniquement utilisateur de ce programme je ne présenterai pas le code.

d. Constatations

Lorsque ce stade d'avancement fut atteint, un constat des résultats obtenus a été nécessaire afin de bien définir les performances et la marge de progression qu'il nous restait. Nous avons donc effectué des tests sur des banques de vidéos fournies par les agents de Nogent-sur-Vernisson. Suite à quoi, nous avons pu comparer nos résultats avec les leurs et prendre contact avec eux afin de fixer les exigences finales de l'algorithme. Les premiers résultats sont donc très concluants et nous avons réussi, à ce stade, à ne rater qu'un faible pourcentage d'animaux de passages dont la marge d'erreurs est comprise entre 5 à 10%. Mais la conséquence est que nous récupérons, en même temps, un nombre important de bruits de l'ordre de 30%. Cela impacte nos résultats négativement. D'où la nécessité d'implémenter une solution complémentaire permettant de garder un taux d'erreurs faible de détection, de l'ordre de 5% maximum, et aussi un taux de bruit quasiment nul idéalement.

B. Couplage au Deep Learning

a. Tensorflow et Imagenet

La suite du travail concerne donc l'implémentation d'une solution de Deep Learning visant à répondre aux attentes du constat fait précédemment. Pour ce faire, dans un premier temps, Bernard Benet et moi-même avons assisté à un séminaire à l'ISIMA (Institut Supérieur d'Informatique, de Modélisation et de leurs Application) **[9]** sur le Deep Learning. Suite à quoi nous avons envisagé de l'utiliser dans notre algorithme. Pour cela nous avons premièrement pris rendez-vous avec des experts (Thierry Chateau et Pierre Yves Lacroix enseignants chercheurs à l'Université Clermont Auvergne) afin d'explorer les différentes options que nous avions. C'est à l'issue de ce rendez-vous que nous avons fait le choix d'utiliser la bibliothèque Tensorflow **[10]**, ainsi que le modèle Imagenet **[11]** entrainé par Google sur des milliers de photos d'espèces animales différentes.

Le Deep Learning permet d'entraîner en amont un algorithme qui serait capable par la suite, lors de son utilisation, de converger rapidement vers un résultat en se basant sur de grands nombres de paramètres d'entrée et avec un taux d'erreur très faible. Ce type de technique n'est pas nouveau mais à gagner en succès récemment grâce à des résultats prometteurs difficilement égalables autrement. Ce type de méthode peut être utilisé dans de nombreux domaines. Un cas d'application classique est le traitement d'image où chaque pixel est un paramètre d'entrée et vient nourrir l'algorithme.

Afin de vérifier si ce type de méthode pourrait être envisageable dans notre cas nous avons utilisé le modèle entrainé par Google appelé Imagenet en envoyant des images extraites de nos cas d'étude. Et il s'est avéré que, même dans les cas complexes, en jouant sur

la couche englobante et sur le format des images (nombres de paramètres en entrée) nous avons réussi à faire réagir positivement le modèle. Nos résultats ont été assez encourageants pour commencer à implémenter cette solution sur notre programme principal. En ce qui nous concerne l'algorithme de Deep Learning permet de signifier, avec un certain pourcentage de chance, la présence d'une espèce animale. Vous trouverez ci-dessous des exemples de performances atteintes grâce au Deep Learning. **Fig. 16. Exemples de résultats utilisant Imagenet.**

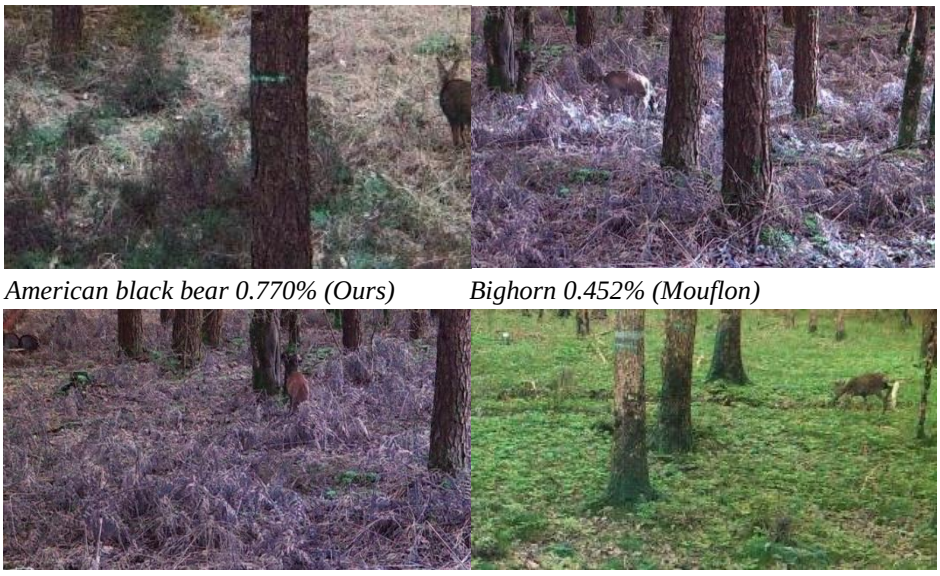


***Figure n°16** : Exemples de résultats utilisant Imagenet*

b. Méthodologie du réseau

Dans le cadre de notre étude nous avons utilisé le Deep Learning en raison de ses nombreuses prouesses dans la science des données de tout genre et surtout dans le domaine de l'image. Plus précisément nous avons fait du « Transfert Learning », cela consiste à réutiliser des réseaux pré-entrainés à d'autres problématiques. Il est donc important de comprendre sur quels principes notre réseau repose afin de pouvoir envisager de le modifier ou d'en entrainer un nous-même. Le Deep Learning a plusieurs champs d'application, dans notre cas il s'agit de la vision par ordinateur. Cet univers de travail est à la base de plusieurs techniques spécifiques d'apprentissage particulièrement efficaces. Nous allons détailler différents principes d'apprentissage utilisés dans le cadre du traitement d'images et ainsi présenter celui utilisé pour notre modèle et les avantages qu'il apporte.

Dans un premier temps, penchons-nous sur la structure globale du réseau. Un réseau de neurones classique peut être schématisé par trois couches, une première couche représentant les entrées connectées à notre source de données (par exemple chaque pixel d'une image), une deuxième couche souvent composée de plusieurs couches qui sont responsables du calcul de la propagation et de la convergence de la dernière couche qui elle,

correspond aux sorties, et donc aux résultats. Chaque couche communique avec la couche adjacente via des connexions entre neurones (neurones composés chacun d'une fonction non linéaire et d'un poids associé, capable d'exciter ou non les neurones correspondants de la couche adjacente). Le but de la création et de l'entrainement de réseau consiste donc au choix du nombre et la mise en place des couches ainsi qu'à la répartition des poids.

Dans le cas du réseau Imagenet que nous utilisons, la méthode mise en place afin de déterminer les poids et choisir les couches est une des méthodes les plus classiques et efficaces dans le cadre du traitement d'images. En revanche ce n'est pas la plus simple à entrainer (couteuse en expertise et en matériel). Ceci pouvant justifier le fait qu'un tel réseau est plus souvent soumis à du Transfert Learning qu'à un réentrainement spécifique. Le principe utilisé est appelé CNN (Convolutional Neural Networks) et est constitué d'une première couche convolutive faite de filtres de convolutions permettant d'extraire des caractéristiques de l'image. La couche suivante appelée perceptron multicouche combine les caractéristiques pour effectuer une classification de l'image, et la dernière couche distribue les probabilités de sortie.

La dernière partie permettant de donner vie à notre réseau et de commencer à l'utiliser est la plus importante. Elle va permettre d'accorder les différents poids de chaque neurone afin de réduire le taux d'erreur au maximum. Cette partie est très spéciale, elle est basée sur la rétropropagation du gradient. Plus précisément, de nombreuses images seront envoyées en entrée afin de créer un cheminement de l'information. Par la suite les résultats sont rétro propagés dans le sens contraire afin de réajuster les poids définis précédemment. Cette opération a pour but de limiter le nombre de couches du perceptron afin de réduire le nombre de calculs inutiles ou répétitifs. Cela permet de converger plus efficacement vers le bon résultat.

Actuellement beaucoup de chercheurs s'efforcent d'inventer de nouvelles méthodes d'apprentissage, ou encore de nouvelles combines afin d'adapter des CNN pré-entrainés. L'extracteur automatique de caractéristique ou encore le fait de ré-entrainer plus finement un réseau (Fine Tuning) sont à titre d'exemple, des pratiques courantes d'opérations effectuées sur des réseaux convolutifs. Dans le cadre de notre étude, il paraitrait intéressant soit d'utiliser le même CNN, mais en l'entrainant avec nos images, soit d'utiliser une autre méthode, par exemple en utilisant des réseaux siamois. Cette technique permettrait d'améliorer nos résultats tout en se basant sur une base de données petite et de bénéficier des avantages de la méthode décrite précédemment. C'est une hybridation entre la classification et l'apprentissage métrique en se basant sur la combinaison du coût de la classification et un coût de minimisation de distance intra-classe qui a pour but de faciliter la classification. Malheureusement ce type de réseaux est encore très rare, le travail de mise en place est donc quasiment entièrement à réaliser.

c. Implémentation

Nous avons donc implémenté le Deep Learning à notre programme. Pour cela nous avons dû transformer notre programme en C++ en un programme exactement équivalent mais en Python. Le modèle fourni par Google est utilisable via un programme en Python. En incluant en python les fonctions nécessaires à l'utilisation du modèle dans notre algorithme de traitement d'image, nous avons mis en place l'environnement.

La méthode utilisée ensuite pour faire intervenir notre nouvelle couche est expliquée ci-dessous. Notre programme permettait jusqu'à présent de ne rater quasiment aucun animal, mais d'avoir, en contrepartie, un taux de bruit élevé. Nos images contenant des animaux étaient quelque peu noyées au milieu "de faux positifs". Pour limiter ces nombres de faux positifs nous avons procéder à la vérification de nos résultats par le Deep Learning. Ainsi, pour chaque label trouvé, une vérification par le Deep Learning vient confirmer la présence de l'animal et, par conséquent, exclure du résultat tous les bruits **Fig. 17. Illustration de l'implémentation du Deep Learning.**

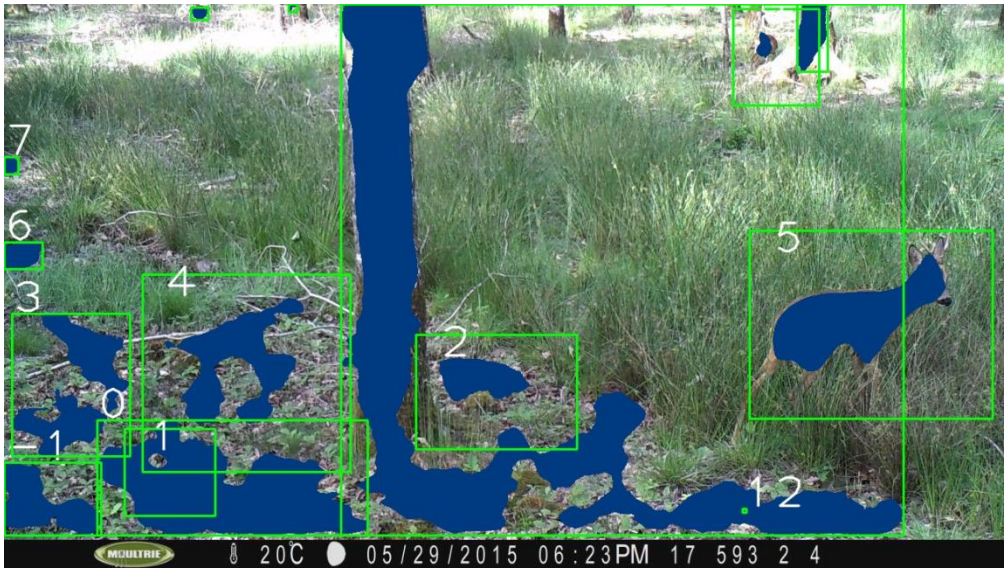


***Figure n°17** : Illustration de l'implémentation du Deep Learning*

Bien évidemment de nombreux problèmes sont apparus et ont nécessité la mise en place d'une phase d'ajustement de la boîte englobante. Par exemple, la base de données ne permet pas de détecter uniquement des animaux. Elle permet aussi de détecter tout type d'éléments présents à l'image. Il a donc fallu extraire du calcul tous les nouveaux faux positifs que le programme pouvait potentiellement détecter, comme les toiles d'araignée ou encore les résultats du type "présence d'une forêt" ou "présence d'arbres" **Fig. 18. Illustration de résultats obtenus avec Imagenet**.

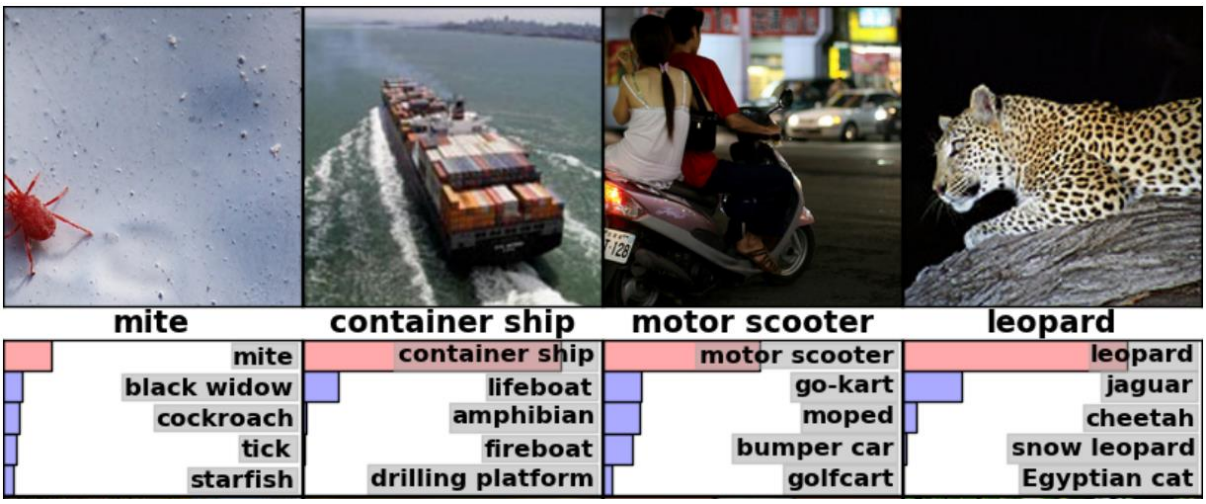


Figure n°18 *: Illustration de résultats obtenus avec Imagenet*

Afin de maximiser les chances de détection, une phase d'opération sur l'image est nécessaire. Si le nombre de labels est très important, dans ce cas nous divisons l'image en sous partie avec recouvrement et redimensionnement. Puis on procède à la même première opération afin d'avoir différentes résolutions d'image. Nous envoyons successivement au Deep Learning toutes les sous images créés. Dans le cas où le nombre de labels détectés est faible, nous envoyons uniquement les parties de l'image encadrant le label avec redimensionnement successif pour les mêmes raisons expliquées précédemment. Cette nouvelle couche de calcul va donc venir confirmer la présence d'animaux et donc limiter considérablement le nombre de faux positifs. Vous trouverez en annexes 8 et 9 les parties du code correspondantes à l'implémentation du Deep Learning, plus précisément, les modifications de la boîte englobante ainsi que les opérations sur les fichiers. Voici, ci-dessous, un exemple de l'affichage utilisé lors de la création du programme. Vous pouvez observer le séquencement, vidéos par vidéos, images par images, puis labels par labels, et ainsi de suite **Fig. 19. Affichage/Debugger**.

```
Image numero =  31
Moyenne de gris = 120.993765191
-Taille avant seuillage = 22
-Taille apres seuillage = 15

label numero = 5
2018-06-06 13:49:09.113510: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU sup
2018-06-06 13:49:09.551858: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithG
.nn.batch_normalization().
hen-of-the-woods, hen of the woods, Polyporus frondosus, Grifola frondosa (score = 0.05804)
vine snake (score = 0.20868)
maze, labyrinth (score = 0.13569)
label numero = 6
hartebeest (score = 0.29350)
ibex, Capra ibex (score = 0.23697)

Image numero =  32
Moyenne de gris = 121.683472222
-Taille avant seuillage = 16
-Taille apres seuillage = 15

label numero = 5
wood rabbit, cottontail, cottontail rabbit (score = 0.10369)
wood rabbit, cottontail, cottontail rabbit (score = 0.04545)
fox squirrel, eastern fox squirrel, Sciurus niger (score = 0.18675)
label numero = 6
```

Figure n °19 *: Affichage/Debugger*

d. Bilan

Suite à l'implémentation de cette solution complémentaire, Bernard Benet et moi-même sommes allés à Nogent-sur-Vernisson afin de présenter notre travail et déterminer ensemble la forme que prendrait la proposition finale. Suite à cet échange, notre solution fut retenue. Notre nouvel objectif était donc la mise en place de notre proposition installée sur une machine fournie par leurs soins et l'optimisation des vitesses de calculs via l'optimisation du calcul par cartes graphiques. A ce stade, il est aussi envisagé l'entrainement d'un modèle sur mesure à notre sujet d'étude. Mais la contrainte du temps qu'il reste ne nous permettra pas d'exploiter cette dernière piste car le Deep Learning a été envisagé tardivement. Cela permettrait une valorisation importante de notre étude dans l'éventualité d'une publication.

C. Elargissement du contexte

Actuellement la surveillance de la biodiversité est un enjeu capital quant à la préservation des espèces animales et végétales, au recensement, à la gestion responsable de l'environnement ainsi qu'à de nombreux nouveaux objectifs apparaissant chaque jour, notamment à cause des changements environnementaux croissants.

Cependant, l'utilisation de solutions automatiques à la collecte de données photographiques dans le domaine de l'écologie, est nouvelle, et a seulement commencé récemment (2012). Jusqu'à maintenant les études photographiques de la faune utilisaient une identification manuelle qui s'avère décourageante pour des grandes bases de données. Ces travaux ont conduit aujourd'hui à des méthodes de reconnaissance assistée par ordinateur, qui sont d'autant plus efficace que les puissances de calculs augmentent exponentiellement depuis quelques années. Les grands enjeux de ce type de recherches visent la réduction de l'allocation mémoire, l'augmentation des performances sur la rapidité d'exécution et la robustesse de détection, la mise en commun et à disposition de solutions nouvelles.

La solution que nous avons mise en place répond aux divers enjeux de ce domaine de recherche. La solution est utilisable dans le cadre de la détection d'animaux via une caméra fixe et dans un milieu forestier. Mais la méthode que nous avons développée est ajustable à n'importe quel autre cas de figure (désert, fond marin, ciel etc…). Cette solution s'inscrit totalement dans le besoin naissant de créer des pièges photographiques efficaces se distinguant d'une détection de type « surveillance de rue ». Ici, notre piège est adapté aux milieux complexes sujets à de nombreux encombrements (luminosité, végétation, occlusion, encombrement). Cette solution est basée sur un couplage de deux méthodes, premièrement un traitement d'image classique et ensuite une confirmation de décision par Deep Learning.

D'après les premiers résultats obtenus sur des échantillons couvrants plusieurs mois de prises photos comportant environ 30 000 images tests, nous avons obtenu des résultats supérieurs à 85% de réussite, pouvant aller jusqu'à 95%. Tenant compte du fait que la solution vient tout juste d'être implémentée, et nécessite de nombreuses améliorations, nos

résultats sont très encourageants et nous poussent à croire que notre solution pourrait être très bénéfique à la recherche scientifique sur un plan international aux vues des différentes publications récentes sur le sujet. **[12] [13] [14]**. Une des publications qui illustre parfaitement mon propos présente le travail d'une stratup américaine posté il y a trois semaines. Cette startup est précurseur d'un algorithme identifiable au notre permettant de détecter les animaux à 96,6% de précision, basé sur le même type de réseau convolutif et Tensorflow, et permettant d'identifier des lamantins. La solution proposée par cette stratup a été accepté et vendu, elle permettra d'après les calculs d'économiser 50 000 postes par ans pour les 15 prochaines années et ainsi économiser en main d'œuvre plusieurs millions d'euros. Aux vues de nos résultats nous pourrions facilement à l'aide du matériel nécessaire et des moyens financiers, crée une solution encore plus performante que celle proposé par cette stratup. **[15]**.

IV. Conclusion

Les techniques classiques de traitement d'images se sont montrées très efficaces, malgré la complexité du milieu. Nous avons réussi à garder un contrôle sur les différents seuils pour atteindre la précision voulue malgré le fort taux de bruit. Ce type de traitement peut donc s'avérer extrêmement efficace en solution unique pour tout autre cas d'étude moins sensible aux variations de l'environnement. La morphologie d'images et les différentes fonctions d'OpenCV offrent la possibilité d'effectuer un grand nombre d'opérations sur l'image de manière optimale et synthétique. De plus, que ça soit le C++ ou le Python, les deux langages sont très bien adaptés à ce type de traitement. Je n'ai eu quasiment aucun mal à passer d'un langage à l'autre. Le C++ offre quand même un meilleur contrôle des données transitant, et théoriquement devrait être plus rapide (mais difficilement remarquable à l'œil nu).

Dans le cas d'un traitement d'images sur un environnement très complexe comme le nôtre, le Deep Learning s'est avéré être très efficace. Nous avons pu éliminer grâce à cette méthode tous les bruits et ne récupérer, en sortie, que les images avec des animaux présents. Bien que les animaux détectés ne correspondent pas exactement à l'espèce « réel sur l'image », cela n'a pas été une contrainte car notre but était uniquement de détecter une présence animale et non l'espèce exacte. De ce fait, il serait intéressant de ré-entraîner le modèle de Imagenet afin de le faire fonctionner sur une base contenant uniquement les cas qui nous concernent et ainsi valoriser un maximum le travail réalisé. De plus, l'étude n'étant pas totalement terminée, beaucoup d'améliorations peuvent encore être faites afin de déterminer l'espèce en question via la boîte englobante, ou encore préparer le livrable en optimisant au maximum les vitesses de calculs.

D'un point de vue personnel, ce stage a été enrichissant. J'ai appris à maîtriser deux langages de programmation, le Python et le C++. N'ayant, auparavant, jamais fait de traitement d'image, j'ai pu acquérir un savoir précieux que je compte réutiliser dans la suite donnée à ma carrière. De même j'ai appris à utiliser le Deep Learning que je n'avais jamais expérimenté par le passé et pour lequel je n'avais jamais eu de formation. Enfin j'ai eu à utiliser l'architecture de programmation ROS (Robot Operating System). Ceci représente une valeur ajoutée importante aux yeux des entreprises. Grâce à cette expérience, j'ai donc eu l'occasion de contribuer au développement de solutions algorithmiques de traitement d'image, précisément dans la détection d'animaux en forêt mais aussi l'opportunité d'intégrer une équipe de recherche qui m'a permis d'affiner mon projet professionnel. Participer à la fois à un projet de recherche, et bénéficier de l'expérience de chercheurs expérimentés fut l'occasion d'éprouver de nombreuses connaissances acquises tout au long de mon cursus.

V. Annexes

```
def split():
  numvid = 8
  while numvid != 9:
    if numvid > 9:
      video = '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/test_video/Plot00'+str(numvid)
      video = video +'-'
      video = video +str(numvid - 1)
      video = video +'.mp4'
    else:
      video = '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/test_video/nogent'+str(numvid)
      #video = video +'-'
      #video = video +str(numvid - 1)
      video = video +'.avi'
    print("* Filename: ", video)
    cap = cv2.VideoCapture(video)
    if not cap:
      printf("!!! cvCaptureFromAVI failed (file not found?)")
      return -1;
    fps = cap.get(cv2.CAP_PROP_FPS)
    fpsi = int(fps)
    frame = 'NULL'
    frame_number = 0
    key = '0'
    while key != 'q' and frame_number < 241:
      frame=cap.read()
      if not frame:
        print("!!! cvQueryFrame failed: no frame")
        break
      filename = '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/frametampon/frame_'+str(frame_number)+'.jpg'
      cv2.imwrite(filename,frame[1])
      frame_number = frame_number + 1
      waitingkey = 1000/fps
      cv2.waitKey(int(waitingkey))
    numvid = numvid + 1
```

Annexe 1 *: Fonction de Splitting des vidéos*

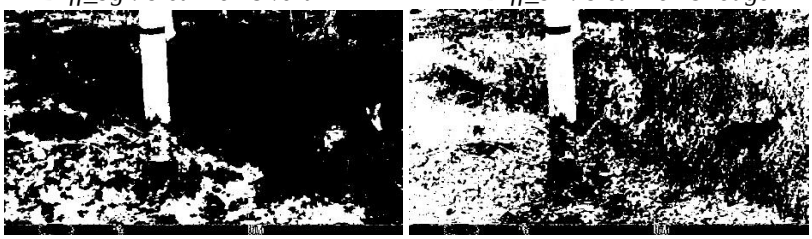
```
def main(_):
  global k
  #create_graph()
  #split()
  #split_for_deep()
  sizeY=720
  sizeX=1280
  seuilmeangray = 40
  seuildifgray = 1
  seuildifcol = 5
  nbr_median = 3
  param_median = 2
  nbr_image_non_traite = 0

  for k in range(31,200):
    strname1="/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/frametampon/frame_"+ str(k) + ".jpg"
    strname2="/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/frametampon/frame_"+ str(k+1) + ".jpg"
    strname3="/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/frametampon/frame_"+ str(k+1) + ".jpg"

    ImageColor1 = cv2.imread(strname1)
    ImageColor2 = cv2.imread(strname2)
    ImageColor3 = cv2.imread(strname3)

    # Calcul de la difference de gris :
    ImageGray1 = cv2.cvtColor(ImageColor1,cv2.COLOR_RGB2GRAY)
    cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/pourpresentation/yolo1.jpg',ImageGray1)
    ImageGray2 = cv2.cvtColor(ImageColor2,cv2.COLOR_RGB2GRAY)
    cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/pourpresentation/yolo2.jpg',ImageGray2)
    ImageDiffGray = cv2.absdiff(ImageGray1, ImageGray2)
    cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/pourpresentation/yolo3.jpg',ImageDiffGray)
    meanGray =  cv2.mean(ImageGray1)

    if meanGray[0] < seuilmeangray :
      nbr_image_non_traite += 1
    else:
      # Binarisation et seuillage :
      ret,gray_tresh = cv2.threshold(ImageDiffGray,1,255,cv2.THRESH_BINARY)
      cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
FevMars2013_premier_Resultats/pourpresentation/yolo4.jpg',gray_tresh)
```

Annexe 2 *: Différence de gris et binarisation + seuillage*

```
# Seuillage des couleurs :
b,g,r = cv2.split(ImageColor2)
Diff_bg = cv2.subtract(b,g)
Diff_br = cv2.subtract(b,r)
Diff_rb = cv2.subtract(r,b)
Diff_rg = cv2.subtract(r,g)

ret1,bg_tresh = cv2.threshold(Diff_bg,0,255,cv2.THRESH_BINARY)
ret2,br_tresh = cv2.threshold(Diff_br,0,255,cv2.THRESH_BINARY)
ret3,rb_tresh = cv2.threshold(Diff_rb,0,255,cv2.THRESH_BINARY)
ret4,rg_tresh = cv2.threshold(Diff_rg,0,255,cv2.THRESH_BINARY)

rb_rg = cv2.bitwise_and(rb_tresh,rg_tresh)
bg_br = cv2.bitwise_and(bg_tresh,br_tresh)
bg_br_rg_rb = cv2.bitwise_or(bg_br,rb_rg)


ImageFinale = cv2.bitwise_and(gray_tresh,bg_br_rg_rb)

cv2.rectangle(ImageFinale,(0,680),(1280,720),0,cv2.FILLED)
cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
evMars2013_premier_Resultats/pourpresentation/yolo5.jpg',ImageFinale)
# Operations morphologiques :
kernel1 = np.ones((3,3), np.uint8)
ImageFinale = cv2.erode(ImageFinale, kernel1)
kernel2 = np.ones((3,3), np.uint8)
ImageFinale = cv2.dilate(ImageFinale, kernel2)
cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/
evMars2013_premier_Resultats/pourpresentation/yolo6.jpg',ImageFinale)
ImageFinale = cv2.medianBlur(ImageFinale,41)
```

Annexe 3 *: Seuillage des couleurs et opérations morphologiques*

Diff_bg : bleu moins vert *Diff_br : bleu moins rouge*



Seuillage de Diff_bg : bg_tresh *Seuillage de Diff_br : br_tresh*

Opération booléenne : (Diff_bg et Diff_br) = bg_br

Annexe 4 *: Détails en image des opérations sur la couleur pour les animaux aux teintes bleutées*

Diff_rb : rouge moins bleu

Diff_rg : rouge moins vert

Seuillage de Diff_rb : rb_tresh

Seuillage de Diff_rg : rg_tresh

Opération booléenne : (Diff_rb et Diff rg) = rb_rg

Opération booléenne finale : (rb_rg ou bg_br) = rb_rg_bg_br

***Annexe 4** : Détails en image des opérations sur la couleur pour les animaux aux teintes rouges*

```
import os.path
import cv2
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

repertoire = os.getcwd()
repertoire_photo =repertoire +"/Img/"
repertoire_arbre = repertoire +"/Arbre/"
numArbre = 0
numPoints = 0

def Apprentissage():
  global repertoire_photo,image,b,numPoints,numArbre
  w, h = 20, 50;
  b = [[0 for x in range(w)] for y in range(h)]
  photo= os.listdir(repertoire_photo)

  for i in range (0, len(photo)):
    image = cv2.imread(repertoire_photo+photo[i],1)
    Refpt=[]
    res = 0
    cv2.imshow("image",image)
    cv2.setMouseCallback("image",click_and_drag,0)
    k=cv2.waitKey(0)
    if(k & 0xFF == 27):
        cv2.destroyWindow("image")

        break

    mon_fichier_arbre = open(repertoire + "/Arbre"+str(i)+".txt","w")
    mon_fichier_arbre.write(str(numArbre)+"\n")
    for i in range (numArbre):
       mon_fichier_arbre.write(str(b[i][0])+ " "+ str(b[i][1])+" "+str(b[i][2])+
[8])+" "+str(b[i][9])+" "+str(b[i][10])+" "+str(b[i][11])+"\n")
    numArbre = 0
    mon_fichier_arbre.close()

def click_and_drag(event,x,y,flags,params):
    global image
    global Refpt,Refpt2
    global res
    global a,numArbre,numPoints
    numARbre = 0
    if(event == cv2.EVENT_LBUTTONDOWN):
        Refpt =x,y
    elif (event == cv2.EVENT_LBUTTONUP):
        Refpt= x, y
        a = np.array([min(Refpt[0],x),min(Refpt[1],y)], int)
        b[numArbre][numPoints] = a[0]
        b[numArbre][numPoints+1] = a[1]
        cv2.moveWindow("ROI",1500,0)
        save = 0
        numPoints = numPoints +2
    photo= os.listdir(repertoire_photo)
    for i in range (0, len(photo)):
      if( numPoints == 12 ):
        numPoints = 0

Apprentissage()
```

Annexe 4 *: Programme d'extraction des arbres*

```
// Annulation des points trouvé sur les Arbres
if(OptionArbre==1)
{
  for(int n=0;n<numArbre;n++)
  {
    Point ppt2[12];
    ppt2[0].x = myvector[0][n];
    ppt2[0].y = myvector[1][n];
    ppt2[1].x = myvector[2][n];
    ppt2[1].y = myvector[3][n];
    ppt2[2].x = myvector[4][n];
    ppt2[2].y = myvector[5][n];
    ppt2[3].x = myvector[6][n];
    ppt2[3].y = myvector[7][n];
    ppt2[4].x = myvector[8][n];
    ppt2[4].y = myvector[9][n];
    ppt2[5].x = myvector[10][n];
    ppt2[5].y = myvector[11][n];
    fillConvexPoly( ImageColor1,ppt2,6,Scalar(255));
    fillConvexPoly( ImageDiffGray,ppt2,6,Scalar(0));
  }
}
```

***Annexe 5** : Utilisation du programme précédent dans le programme principal*

```
// Recherche des objets séparés et de leurs contours
vector<vector<cv::Point> > contours;
cv::findContours(ImageDiffGray, contours,  CV_RETR_LIST, CV_CHAIN_APPROX_NONE);

// Déclaration des variables (points de contour, rectangle,...)
vector<vector<cv::Point> > contours_poly( contours.size() );
vector<cv::RotatedRect> minRect ( contours.size() );

// Balayage des contours et obtention des rectangles, cercles, ellipses,...
for (size_t m = 0; m < contours.size(); m++ )
{
  // Rectangles circonscrits aux objets
  approxPolyDP( cv::Mat(contours[m]), contours_poly[m], 3, true );

  // Rectangles plus précis (orientation) circonscrits aux objets
  minRect[m]=cv::minAreaRect(cv::Mat(contours_poly[m]));
  cv::Point2f rect_points[4];
  minRect[m].points( rect_points );

  // Paramètres géométriques de chaque objet
  LargeurRecInscrit[m]=minRect[m].size.width;
  LongueurRecInscrit[m]=minRect[m].size.height;
  Elongation[m]=CalculElongation(LargeurRecInscrit[m],LongueurRecInscrit[m]);
  //std::cout<<"ELongation= "<<Elongation[m]<<endl;
  Point ppt[4];
  ppt[0] = rect_points[0];
  ppt[1] = rect_points[1];
  ppt[2] = rect_points[2];
  ppt[3] = rect_points[3];

  // Annulation des objets trouvé par un seuil de leurs élongation
  if ( Elongation[m]>seuilelongation )
    fillConvexPoly( ImageDiffGray,ppt,4,Scalar(0));
}
```

Annexe 6 *: Elimination des labels aberrants par calcul d'élongation*

```
      # Rangement des labels trouvés :
      im2, contours, hierarchy = cv2.findContours(ImageFinale, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
      print('-Taille avant seuillage =',len(contours))

      # Pour chaque labels, seuillage :
      for i in range(len(contours)):
        cnt = contours[i]
        area = cv2.contourArea(cnt)
        rect = cv2.minAreaRect(cnt)
        y = (rect[0][1]+(rect[0][1]-rect[1][1]))/2

        if y<730 and y>=400:
          seuillabelmin=500
        if y<400 and y>=200:
          seuillabelmin=200
        if y<200 and y>=-60:
          seuillabelmin=20

        # Elimination des labels ne réspectant pas les seuils :
        if area < seuillabelmin:
          ImageFinale = cv2.drawContours(ImageFinale,[cnt],-1,0,-1)
```

Annexe 7 *: Elimination des labels aberrants par seuillage*

```
def run_inference_on_image(image):
  """Runs inference on an image.
  Args:
    image: Image file name.
  Returns:
    Nothing
  """
  global k

  if not tf.gfile.Exists(image):
    tf.logging.fatal('File does not exist %s', image)
  image_data = tf.gfile.FastGFile(image, 'rb').read()

  # Creates graph from saved GraphDef.
  create_graph()

  with tf.Session() as sess:

    softmax_tensor = sess.graph.get_tensor_by_name('softmax:0')
    predictions = sess.run(softmax_tensor,
                           {'DecodeJpeg/contents:0': image_data})

    predictions = np.squeeze(predictions)

    # Creates node ID --> English string lookup.
    node_lookup = NodeLookup()
    top_k = predictions.argsort()[-FLAGS.num_top_predictions:][::-1]
    human_string = node_lookup.id_to_string(top_k[0])
    score = predictions[top_k[0]]
    res = 0

    if (score > 0.20 and human_string != 'worm fence, snake fence, snake-rail fence, Virginia fence' and human_string != 'cliff, drop, drop-off' and human_string != 'mountain
bike, all-terrain bike, off-roader' and human_string != 'pole' and human_string != "spider web, spider's web" and human_string != 'stole' and human_string != 'viaduct' and
human_string != 'hen-of-the-woods, hen of the woods, Polyporus frondosus, Grifola frondosa' and human_string != 'milk can' and ( human_string == 'water ouzel, dipper' or
human_string == 'quail' or human_string== 'dhole, Cuon alpinus' or human_string =='ibex, Capra ibex' or human_string =='redbone' or human_string =='ox' or human_string
=='Brabancon griffon' or human_string =='impala, Aepyceros melampus' or human_string =='hare' or human_string =='wallaby, brush kangaroo' or human_string =='wombat' or
human_string =='weasel' or human_string =='gazelle')) or (score > 0.35 and human_string == 'fox squirrel, eastern fox squirrel, Sciurus niger') :
        CSI='\x1B['
        print(CSI+"31;40m" +'%s (score = %.5f)'% (human_string, score) + CSI + "0m")
        strname = "/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars2013_premier_Resultats/frametampon/frame_"+ str(k+1) + ".jpg"
        img2 = cv2.imread(strname)
        cv2.imwrite('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars2013_premier_Resultats/resultats_deep3/'+str(k+1)+human_string+'_'+str(score)+'.jpg',img2)
        res = 1
      else:
        print('%s (score = %.5f)' % (human_string, score))
    return res
```

Annexe 8 *: Modification de la boîte englobante*

```
# Pour chaques labels ou contours :
for i in range(len(contours1)):
    print('label numero =', i)
    cnt2 = contours1[i]
    x,y,w,h = cv2.boundingRect(cnt2)
# Redimenssionement et création des images déstinée au deep learning :
    if x>30 and x<1250 and 30<y and y<650 and (x+w)<1250 and (y+h)<640 :
      crop0 = ImageColor2[y-30:(y+h)+60, x-30:(x+w)+60]
      crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
      crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
    else:
      if y<30 and x>30 and x<1250:
        crop0 = ImageColor2[y:(y+h)+60, x-30:(x+w)+60]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
      elif y>650 and x>30 and x<1250:
        crop0 = ImageColor2[y-30:(y+h), x-30:(x+w)+60]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
      elif y>30 and y<650 and x<30:
        crop0 = ImageColor2[y-30:(y+h)+60, x:(x+w)+60]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
      elif x>1250 and y>30 and y<650:
        crop0 = ImageColor2[y-30:(y+h)+60, x-30:(x+w)]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
      elif x>1250 and y<30:
        crop0 = ImageColor2[y:y+30, x:x+30]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
      else:
        crop0 = ImageColor2[y:(y+h), x:(x+w)]
        crop1 = cv2.resize(crop0,(int(1280),int(700)), interpolation = cv2.INTER_AREA)
        crop2 = cv2.resize(crop1,(int(80),int(40)), interpolation = cv2.INTER_AREA)
    maybe_download_and_extract()
    cv2.imwrite( '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
)
    cv2.imwrite( '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
)
    cv2.imwrite( '/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
)
    image0 = (FLAGS.image_file if FLAGS.image_file else
    os.path.join('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
+'.jpg'))
    image1 = (FLAGS.image_file if FLAGS.image_file else
    os.path.join('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
+'.jpg'))
    image2 = (FLAGS.image_file if FLAGS.image_file else
    os.path.join('/home/administrateur/Bureau/Ressources_Animaux/Video_Nogents/FevMars20
+'.jpg'))
    # Execution du deep learning :
    res = run_inference_on_image(image0)
```

Annexe 9 *: Opération sur les fichiers*

VI. Références

[1] : L'institut | Irstea [Internet]. [cité 20 juin 2018]. Disponible sur: http://www.irstea.fr/linstitut

[2] : TSCF | Irstea [Internet]. [cité 20 juin 2018]. Disponible sur: http://www.irstea.fr/la-recherche/unites-de-recherche/tscf

[3] : Robotique et mobilité pour l'environnement et l'agriculture | Irstea [Internet]. [cité 20 juin 2018]. Disponible sur: http://www.irstea.fr/la-recherche/unites-de-recherche/tscf/robotique-mobilite-environnement-agriculture

[4] : EFNO | Irstea [Internet]. [cité 20 juin 2018]. Disponible sur: http://www.irstea.fr/la-recherche/unites-de-recherche/efno

[5] : OPTMix – Oak Pine Tree Mixture [Internet]. [cité 20 juin 2018]. Disponible sur: https://optmix.irstea.fr/?page_id=25

[6] : poster_inauguration_faune.pdf [Internet]. [cité 20 juin 2018]. Disponible sur: https://optmix.irstea.fr/wp-content/uploads/2015/09/poster_inauguration_faune.pdf

[7] : Computer Vision Courses [Internet]. [cité 20 juin 2018]. Disponible sur: http://chateaut.fr/index.php/teaching/computer-vision-courses

[8] : OpenCV library [Internet]. [cité 20 juin 2018]. Disponible sur: https://opencv.org/

[9] : ISIMA – Institut Supérieur d'Informatique, de Modélisation et de leurs Applications - [Internet]. [cité 20 juin 2018]. Disponible sur: https://www.isima.fr/accueil-english/

[10] : TensorFlow [Internet]. TensorFlow. [cité 20 juin 2018]. Disponible sur: https://www.tensorflow.org/

[11] : ImageNet [Internet]. [cité 20 juin 2018]. Disponible sur: http://www.image-net.org/

[12] : Yu X, Wang J, Kays R, Jansen PA, Wang T, Huang T. Automated identification of animal species in camera trap images. EURASIP Journal on Image and Video Processing [Internet]. déc 2013 [cité 25 juin 2018];2013(1). Disponible sur: https://jivp-eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2013-52

[13] : Weinstein BG. MotionMeerkat: integrating motion video detection and ecological monitoring. Dray S, éditeur. Methods in Ecology and Evolution. mars 2015;6(3):357-62.

[14] : Rowcliffe JM, Field J, Turvey ST, Carbone C. Estimating animal density using camera traps without the need for individual recognition. Journal of Applied Ecology. août 2008;45(4):1228-36.

[15] : Researchers develop AI that identifies and counts wildlife with 96.6% accuracy | VentureBeat [Internet]. [cité 26 juin 2018]. Disponible sur: https://venturebeat-com.cdn.ampproject.org/v/s/venturebeat.com/2018/06/05/researchers-develop-ai-that-identifies-and-counts-wildlife-with-96-6-accuracy/amp