Rapport de projet de fin d'étude

Lidar terrestre et estimation du volume d'arbres en forêt : évaluation des algorithmes du logiciel Computree

Date : 15/02/13 au 15/06/13
Soutenance du Projet de fin d'étude à Cluny le 25/06/13

Elève Arts et Métiers Paritech:	**Lucas Esclatine**
Professeur encadrant Arts et Métiers Paritech :	**Louis Denaud**
Ingénieur encadrant LaboMap :	**Michael Krebs**
Tuteur de stage IGN :	**Jean Christophe Herve**
Encadrant ONF :	**Alexandre Piboule**

Remerciements

Pour la réalisation de ce stage, je tiens à remercier :

-Jean Christophe Hervé pour son accueil et son encadrement durant l'ensemble du stage.

-Mon encadrant principal, Alexandre Piboule, pour m'avoir apporté son soutien, et ses conseils durant l'ensemble de la durée de mon stage.

-Louis Denaud, mon professeur encadrant aux Arts et Métiers Paritech pour son suivi et son enthousiasme dans mes projets.

-Michael Krebs ingénieur au LaboMaP, sans qui Computree ne serait que l'ombre du logiciel qu'il est à présent.

-Jean-Pierre Renaud 02 pour son aide dans la maitrise des outils statistiques et sa bonne humeur permanente.

- Nicolas Gomez, Noémie Pousse, Pascal George, Anne Jolly, Myriam Legay, Mhedi Ladjal pour l'ensemble des agréables moments et discussions enrichissantes que j'ai pu passer avec les chercheurs du pole R&D de Nancy de l'ONF.

-Myriam Legay pour m'avoir accueilli au sein du pôle.

-Aux valeureux stagiaire de l'ONF, Alisée Privat, Edouard Dapoigny et Nuria Sanchez Lopez pour tous les moments passés avec eux.

Sommaire

Introduction

Dans le cadre de leur parcours aux Arts et Métiers ParisTech, les élèves en dernière année ont l'opportunité de réaliser un projet d'expertise. C'est ainsi l'occasion pour les futurs ingénieurs de se consacrer entièrement à un projet de recherche les aidant à se familiariser avec le monde professionnel.

L'Office National des Forêts et le centre des Arts et Métiers ParisTech de Cluny collaborent depuis 2010 sur le développement de Computree. Ce logiciel détecte et mesure, depuis un nuage de points 3D, les arbres d'une placette forestière. J'ai ainsi eu la chance de contribuer modestement au développement de Computree en évaluant les performances de celui-ci sur un jeu de données fournis par l'IGN.

Ce rapport présente le travail effectué dans le cadre de mon projet d'expertise. Dans un premier temps l'ensemble des acteurs du projet Computree ainsi que la problématique de stage sont présentés. Puis dans une seconde partie le fonctionnement de Computree est décrit. Ensuite le jeu de données fourni par l'IGN ainsi que des indicateurs caractérisant l'efficacité du logiciel sont présentés. Finalement les résultats de l'évaluation des performances de Computree sont commentés.

Les termes techniques précédés d'une * sont tous définis dans le lexique à la fin du rapport.

I. Enjeux du stage

Afin de fournir aux lecteurs les informations nécessaires à une bonne compréhension de l'enjeux du stage, une présentation sommaire des différents acteurs est réalisée. Puis dans un deuxième temps la problématique du stage est exposée.

1)Les acteurs

a)Les Arts et Métiers ParisTech

Les Arts et Métiers ParisTech font partie des « grandes écoles d'ingénieurs » françaises. Elle forme plus de 1000 ingénieurs généralistes par an dans les disciplines du génie mécanique, du génie énergétique et du génie industriel. Vieille de plus de deux cents ans, elle dispose de 8 centres répartis sur l'ensemble du territoire français. Les élèves en dernière année ont l'opportunité de se spécialiser et ainsi d'intégrer le centre qui s'est investi dans la recherche et le développement du domaine en question.

Figure 1 : Centre des Arts et Métiers ParisTech Cluny

Le centre de Cluny dispose ainsi de 3 spécialités, dont une consacrée au matériau bois. L'objectif de cette spécialité est de fournir aux futurs ingénieurs des outils pour travailler tant en bureau d'étude, qu'en laboratoire de recherche bois, ou que dans les entreprises de première et deuxième transformation du bois.

Le centre des Arts et Métiers de Cluny via le LaboMaP est donc naturellement investi dans cet échange avec l'Office National des Forêts et l'institut de l'information géographie et forestière pour le développement du logiciel Computree.

b) LaBoMaP

Le LaBoMaP est le Laboratoire Bourguignon des Matériaux et Procédés. Il axe ses recherches sur les domaines de la coupe de matériaux et est organisé en trois équipes (l'usinage grande vitesse, les matériaux et usinage bois et matériaux). Concernant le matériau bois l'ensemble des recherches sont dirigées principalement vers le développement de technologies pour valoriser les bois de qualité secondaire ou à croissance rapide dans des produits d'ingénierie bois. La mission principale que s'est fixé l'équipe bois du LaBoMaP concerne donc les bois locaux de faibles diamètres provenant d'éclaircies, mal conformés ou de brins de taillis ainsi que des essences à forte croissance comme le douglas et le peuplier.

Le laboratoire est intégré aux ateliers et locaux de l'école, et une grande partie des enseignants chercheurs du centre des Arts et Métiers de Cluny sont membres permanents du LaboMap. De même une partie des étudiants en thèse font partie des membres non permanents. L'école met ainsi à disposition ses ateliers et locaux pour soutenir les projets entrepris par le LaboMap. Cette collaboration forte entre le LaBoMaP et le centre des Arts et Métiers Paritech est profitable aux deux parties et favorise la recherche et le développement de technologie dans le domaine du bois, des matériaux et de l'usinage grande vitesse.

Michael Krebs est l'acteur principal du LaBoMaP qui contribue au projet Computree depuis son origine. Ingénieur en informatique, il a développé une grande partie des fonctionnalités, notamment le cœur de Computree ainsi que le système de gestion général. De plus il a participé pour une grande partie à la mise en place de l'architecture du logiciel. La politique de développement de la plateforme Computree et de fournir un logiciel avec un maximum de fonctions intégrées de base, mais aussi d'offrir la possibilité à d'autres équipes de recherche de pouvoir implémenter leurs propres algorithmes rapidement et efficacement.

c) L'Institut national de l'information géographique et forestière

L'Institut géographique national (IGN) a fusionné avec l'Inventaire forestier national (IFN) le 1er janvier 2012 pour devenir l'Institut national de l'information géographique et forestière. Le nouvel institut a dans ses missions l'inventaire permanent des ressources forestières en France métropolitaine, auparavant dévolu à l'IFN . La connaissance de la ressource permet aux acteurs économiques de la filière de se développer en pratiquant une gestion durable de la ressource forestière.

C'est donc bien dans le cadre de l'amélioration des inventaires forestiers que l'IGN s'investit dans le projet Computree. Plus précisément Jean-Christophe Hervé, conseiller scientifique pour les méthodes d'inventaire de l'IGN, souhaitait disposer d'une évaluation des performances de Computree. Ceci afin d'évaluer le potentiel de cette plateforme dans le

domaine de l'inventaire forestier.

d) Le pôle de Recherche et Developpement de Nancy de l'Office National des Forêts.

L'office national des forêts est l'organisme public en charge de l'ensemble des forêts publiques du territoire français. Les forêts publiques représentent 24% des forêts françaises et s'étendent sur 4,1 millions d'hectares selon le rapport « La forêt en chiffre et en cartes » 2012 de l'IGN. La mission principale de l'ONF est la gestion durable et multifonctionnelle des forêts publics. En d'autres termes l'ONF rédige et met en œuvre des aménagements tenant compte des fonctions de production, biodiversité, écologie et sociale. Un aménagement, d'une durée de 15 à 20 ans est le document garantissant la gestion durable d'une forêt. Il fourni un bilan de la gestion passé, une analyse des enjeux, un inventaire des peuplement. Pour chaque parcelle, il fixe une essence objectif, ainsi qu'une planification des coupes et travaux sylvicole pour la période

Afin d'optimiser les outils de gestion et de développer des technologies innovantes, l'ONF dispose d'un département recherche et développement comptant environ 70 chargés R&D et assistants R&D réparti en 9 pôles. Les axes principaux de recherche du département R&D sont :

- L'adaptation de la gestion forestière au changement climatique.
- L'accroissement de la disponibilité et l'utilisation de la biomasse forestière pour les produits à base de bois et l'énergie.
- L'élargissement de la gestion durable multifonctionnelle.
- Le développement de produits et procédés innovants pour des marchés et demandes sociales en évolution.

Cette organisation permet de spécialiser chacun des pôles sur des problématiques précises tout en assurant une répartition des pôles de recherches sur l'ensemble du territoire français. Ainsi le pôle de Nancy est en charge des problématiques de changement climatique, télédétection et de la sylviculture des hêtraies continentales. Ainsi, Alexandre Piboule, Chargé de R&D est responsable des projets de recherches portant sur le lidar terrestre, c'est la raison pour laquelle il a entrepri de développer le logiciel Computree.

2) Thème du stage

a)Le LiDAR Terrestre

Le T-LiDAR (Terrestrial Light Detection And Ranging) est un appareil de télédétection. Cet outil laser numérise la géométrie d'une scène à 360° autour de lui. Pour mesurer les coordonnées d'un point, l'appareil balaye l'espace autour de lui, suivant une rotation selon l'axe vertical et une deuxième rotation selon l'axe horizontal. Pendant ces rotations le lidar émet un rayon laser infrarouge qui, s'il est intercepté par un objet, est rétrodiffusé vers le lidar. Le temps entre l'émission et la réception du rayon ainsi que le décalage de phase permettent de déterminer la distance du point et en mesurant les deux angles de l'appareil, le T-LiDAR enregistre les coordonnées sphériques du point. La géométrie de l'appareil ne permet pas de mesurer la totalité de l'espace autour de lui,

Figure 2 : Photographie de lidar terrestre

Figure 3 : Nuage de point 3D visualisé sur Computree

ainsi un cône de 40° en dessous de l'appareil n'est pas mesuré. La réflectance, un indicateur d'intensité du rayon de retour, est aussi mesurée pour chacun des points.

Les domaines d'utilisation de cette technologie sont très divers. Par exemple la préservation du patrimoine culturel, l'architecture, les industries énergétiques, l'aérospatiale, l'automobile, la construction navale, le contrôle qualité automatique, les fonderies, la criminologie, les industries minières. On rencontre le T-LiDAR dans tout les domaines nécessitant une reconstitution précise d'une scène en 3 dimensions.

Dans le contexte d'inventaire forestier, le lidar terrestre est amené directement sur les placettes forestières et scan la végétation et le sol depuis le centre de la placette. Le nuage de point 3D reproduit ainsi fidèlement la géométrie du sol et de la végétation de la placette, il peut ensuite être exploité grâce à Computree.

C'est naturellement que le T-LiDAR suscite un fort intérêt en recherche et développement pour les inventaires forestiers. Cet appareil devrait permettre une mesure très complète et détaillée de la géométrie des placettes forestières*. Cependant la complexité de l'environnement forestier génère certaines difficultés dans le traitement des données, comme l'identification du sol plus ou moins irrégulier, la détection des arbres et de leur appareil aérien, l'occultation générée par la végétation basse. Ainsi depuis une dizaine d'année de nombreux travaux portant sur l'extraction de données dendrométriques* depuis le nuage de point 3D mesuré par le T-LiDAR on déjà été menés dans des équipes de recherche du monde entier. M.Dassot T.Constant M.Fournier ont rédigé un exellent état de l'art portant sur le domaine du lidar terrestre en foresterie, « the use of terrestrial LiDAR technology in forest science : application fields, benefits and challenges ».

L'utilisation du T-Lidar pour la numérisation de placette a cependant certaines limites. Premièrement par temps de vent, les mouvements des branches de faibles diamètres génèrent des décrochages dans le nuage de points 3D. Deuxièmement l'occultation produite par les feuilles et la végétation basse réduit fortement les performances des algorithmes de reconstitutions de la structure des arbres.

b) Problématique

L'IGN a pour mission de réaliser l'inventaire national des ressources forestières et de mettre ses résultats à disposition des décideurs politiques et des acteurs économiques. Les principales informations fournies sont la superficie des forêts et le volume de bois disponible, classé selon divers critères (département, région, type de structure forestière*, type de propriétaire, essence, catégorie de dimension…). Pour cela l'IGN réalise des inventaires statistiques sur un échantillons de 7000 placettes forestières par an environ. Le choix des placettes inventoriées est aléatoire de manière à estimer sans biais la ressource forestière.

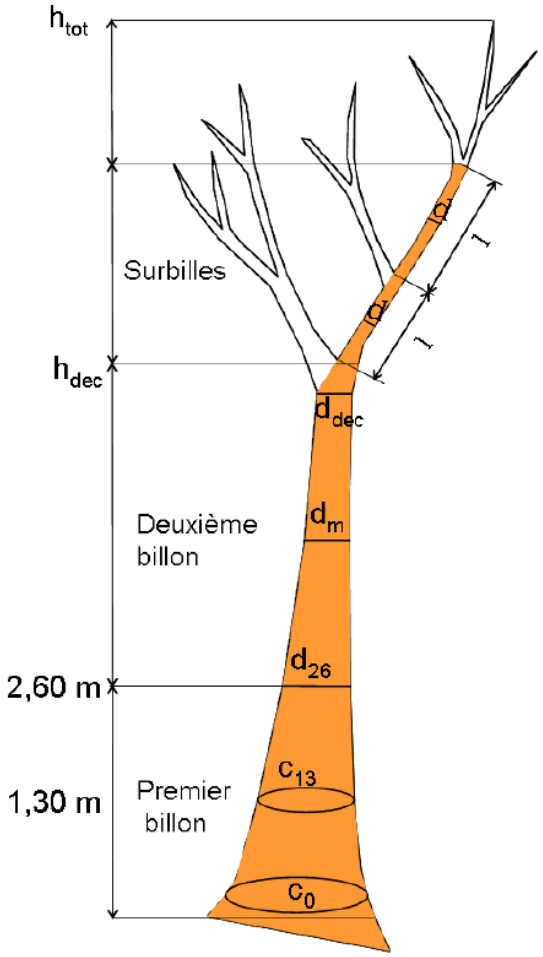


Figure 4 : Schéma de cubage de l'IGN

Le volume de bois mesuré est le volume de la tige principale, arrêté à la limite « bois fort » (7cm de diamètre). La mesure de ce volume (cubage) est réalisé, l'arbre étant sur pied, en décomposant la tige principale en différents billions dont les longueurs et diamètres sont mesurés, comme indiqué sur la figure 4. Un volume est alors calculé. Cependant cette mesure est couteuse en temps, c'est pourquoi l'IGN a mis en place des tarifs* pour chaque espèce, permettant d'estimer le volume bois fort tige grâce aux seules mesures de la circonférence à 1m30 et de la hauteur total. Ces modèles statistiques permettent une estimation précise et non biaisée des volumes à l'échelle régionale ou nationale. Cependant bien que le volume tige soit le volume conventionnellement utilisé pour quantifier la ressource forestière au niveau international, il n'est pas suffisant pour quantifier le bois énergie et pour établir des bilans carbone, car le volume des branches (qui représente en France, de l'ordre de 50% du volume tige) n'est pas comptabilisé.

L'IGN s'est associé au projet Computree afin de disposer à termes d'un outils permettant d'estimer précisément le volume complet d'un arbre, mais aussi d'être capable d'associer aux différentes parties de l'arbre les volumes correspondant. Cela permettra dans un premiers temps d'améliorer les modèles statistiques et donc d'être plus précis sur les tarifs. Dans un deuxième temps cela permettra de fournir des données sur le volume branche en plus des tiges de les mettre à jour dans le futur. Finalement cela offrira la possibilité d'établir des modèles statistiques de l'estimation de volume moyen de bois depuis des relevés classiques pour différentes conventions de volumes. Et ainsi de mettre en place des tarifs adaptés aux différentes demandes des utilisateurs de données d'inventaire.

L'IGN a donc pour objectif de constituer un jeu de donnée d'environ 2000 placettes numérisées au lidar terrestre et représentatif de tout type de peuplement de France. Cette campagne a été lancée en 2009, elle comprend déjà 370 placettes numérisées et est utilisé dans le cadre de ce projet pour réaliser l'évaluation des performances

de Computree. A terme c'est ce jeu de données qui permettra d'améliorer les modèles statistiques et de fournir les estimations de divers volumes de bois.

Ainsi avant d'utiliser Computree sur la totalité du jeu de donnée, il a été choisie dans un premier temps de valider la capacité de Computree à estimer correctement le diamètre à 1 m30 et d'évaluer le taux de détection en fonction de critère comme la structure forestière et la surface terrière de la placette.

II. Computree

1) Introduction

L'étude de mon stage portant sur l'évaluation des performances de Computree, il est essentiel pour la bonne compréhension de ce rapport de décrire plus en détail ce logiciel. Après un rapide historique, la structure et les fonctionnalités principales de Computree sont présentées. Enfin une explication des étapes de traitements du nuage de points est fait.

2) Historique de Computree

Computree a vu le jour en 2010 lors d'un projet de fin d'étude d'un étudiant des Arts et Métiers, Etienne Tricot. A l'époque le projet consistait à rechercher des algorithmes permettant la reconstitution d'inventaires forestiers en utilisant des scans lidar terrestre. Un premier prototype fut développé en java, cependant la taille des nuages de points entrainait des temps de calculs trop long. C'est pourquoi la première version de Computree fut donc codée en C++ avec le framework Qt. Le langage C++ ayant la particularité d'être très rapide cela a permis de réduire de manière importante les temps de calcul. A l'achèvement du stage, Computree contenait déjà les fonctionnalités suivantes :

- Lecture d'un fichier nuage de points
- Séparation en couches et détection de formes
- Détection et Filtrage de cercles
- Détection et Filtrage d'arbres
- Reconstitution de tranches d'arbres.

Puis en 2011 une étudiante en Master 2 FAGE, Aurélie Colin a fait une première évaluation de Computree sur plusieurs jeux de données. Ce stage a permis d'établir que la détection d'arbres était fiable mais que l'estimation du diamètre à 1m30 générait des résultats aberrant dans un nombre de cas non négligeable.

Computree est toujours en phase de développement, avec le suivi régulier et constant de M.Krebs et A.Piboule. La version 2.0 du logiciel est stable et intègre un certain nombre de fonctionnalités décrites dans les parties suivantes.

3) La Structure de Computree

Computree est développé pour être une plateforme collaborative, et se veut un logiciel visuel et modulaire sous la forme d'un laboratoire virtuel. Le logiciel est donc découpé en modules indépendants les uns des autres pouvant communiquer entre eux. La figure 5 présente les différents modules :

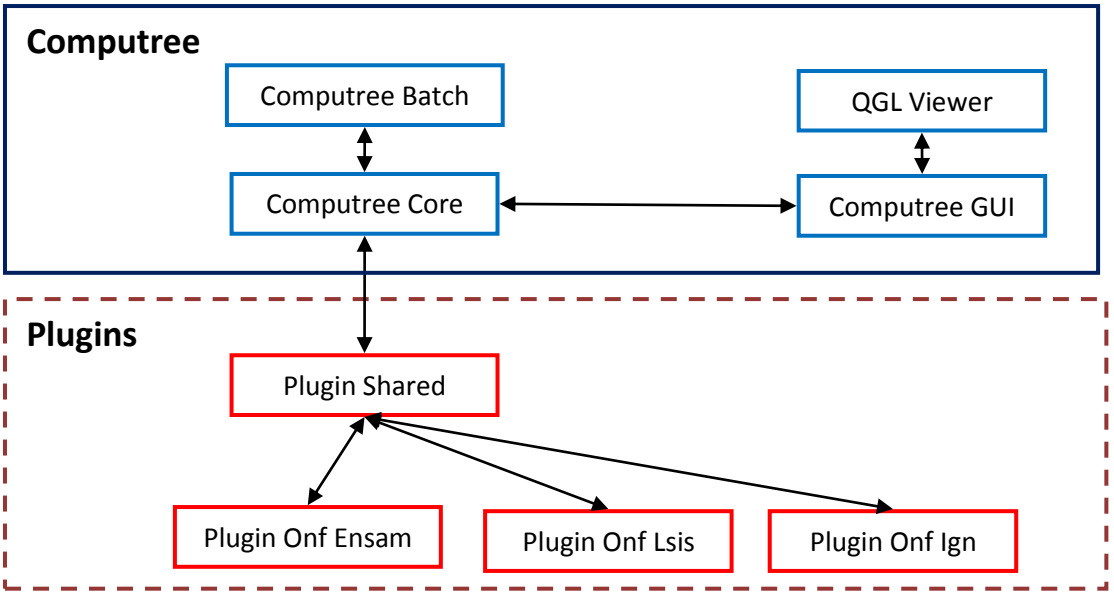


Figure 5 : Schéma de la structure de Computree

Computree GUI, l'interface graphique que l'utilisateur aura à sa disposition pour dialoguer avec le logiciel, est reliée à Computree Core. Computree Core, lui, gère le dialogue entre les différents modules. Quand à QGL Viewer, c'est une librairie préexistante qui permet de visualiser le nuage de point ainsi que d'autres éléments géométriques et de les manipuler en 3D. Ce module donne un aspect graphique très intéressant, offrant la possibilité d'avoir un rendu visuel et de mieux comprendre les résultats des algorithmes. Ces trois premiers modules forment le cœur du logiciel et permettent l'ajout de plugins. S'ajoute Computree Batch qui a un rôle similaire à Computree GUI à l'exception du fait qu'il permet d'exécuter un fichier script préenregistré. Un script est un fichier xml qui décrit l'enchainement des étapes de traitement et les paramètres associés. Une partie du stage fut consacré à étendre la lecture d'un scripte d'un fichier scan à tout les fichiers scans 3D d'un dossier. Ainsi Computree Batch permet un traitement par lot d'un grand nombres de fichiers et ne nécessite pas la présence continue d'un utilisateur. Cela a donc permis d'exécuter le même script sur l'ensemble du jeu de données ciblé de manière automatique. Le code du plugin d'automatisation de la lecture d'un script sur plusieurs fichiers est disponible en annexe II.

Le plugin principal est PluginShared. Il s'agit d'une collection d'objet C++ mis à disposition afin de faciliter l'implémentation d'algorithmes dans Computree pour toute les personnes voulant développer leur Plugin. En d'autres termes ce Plugin fait le lien entre les autres plugins et le cœur de Computree.

Le plugin ONF Ensam contient un ensemble d'étapes de calcul permettant, entre autres, de détecter les arbres et d'estimer le diamètre à 1m30. Le plugin ONF IGN a été développé au cours de ce stage, il calcule automatiquement un certain nombre d'indicateurs placettes et arbres, ces indicateurs seront décrits en détails dans la partie 4.

Il existe aussi d'autres plugins en cours de développement :

- Le Plugin Onf Lsis développé par Joris Ravaglia étudiant au LSIS qui permet de filtrer le nuage de point.
- Le Plugin MeshLab permettant d'utiliser les plugins MeshLab du logiciel de maillage 3D, MeshLab.
- Le Plugin Lvsx, développé par l'université de Sherbrooke, contenant des méthode de voxelisation.

Computree est donc conçu de manière modulaire. Cette structure permet une résolution des bugs plus efficace et permet de se concentrer sur le développement d'un module à la fois. Cette structure présente aussi le grand

avantage de proposer un logiciel avec un certain nombre de fonctionnalités et ainsi de pouvoir implanter facilement de nouveau plugins. Ceci à pour but de créer une synergie entre les équipes de recherche, leurs offrant la possibilité de collaborer par l'intermédiaire de computree.

4) Les fonctionnalités principales de Computree

Dans cette partie, les fonctionnalités principales de Computree vont être présentées afin de décrire en détail les étapes de traitements dans lequel sont faits la détection et l'estimation des diamètres à 1m30. Concrètement l'interface graphique de Computree est divisé en 3 zones (représentée sur la figure 6) :

- Un gestionnaire d'étape permet de visualiser l'enchainement d'étapes, d'en ajouter ou d'en supprimer. Les étapes sont implémentées dans les plugins. Elles contiennent les algorithmes de traitement. Le gestionnaire permet de visualiser la liste des résultats engendrés par les étapes. Le rôle des étapes sera défini par la suite.
- Une zone graphique permet de visualiser et de manipuler le nuage de point et les éléments géométriques calculés dans les étapes.
- Une zone de visualisation des résultats et de leur structure.

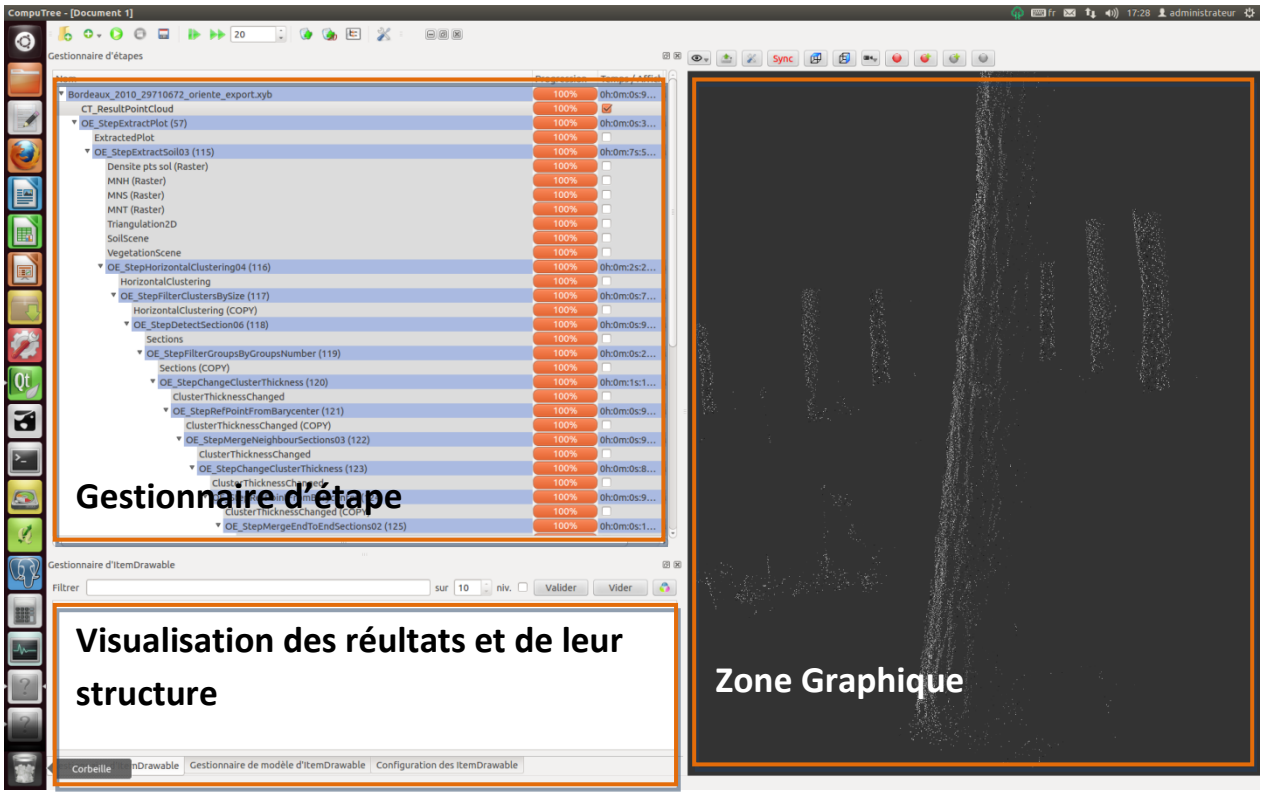


Figure 6 : Interface graphique de Computree

La première fonctionnalité importante de Computree est l'import de fichiers de scans. Un fichier de scan contient l'ensemble des coordonnées du nuage de points. Computree sait importer des fichiers de types xyb et csv.

Une fois le fichier importé dans computree, il est possible de visualiser le nuage 3D dans la zone graphique, en sélectionnant le résultat dans le gestionnaire d'étape. Le résultat de cette première étape d'import est une CT_Scene. Une CT_Scene l'objet disponible dans PluginShared dédié au stockage d'un nuage de points complet. A ce stade l'utilisateur a le choix d'ajouter n'importe quelle étape de n'importe quel Plugin acceptant en entrée un objet CT_Scene. De manière générale une étape génère un résultat et un modèle de sortie. Le résultat est calculé quand l'utilisateur exécute les étapes de calcul, le modèle de sortie lui, est généré immédiatement après l'ajout

de l'étape. Ainsi il est possible d'ajouter plusieurs étapes et de toutes les calculer d'un coup. De plus pour ajouter une étape il faut obligatoirement que le modèle d'entrée de l'étape à ajouter soit compatible avec le modèle de sortie de l'étape antérieure.

Les modèles sont définis par une arborescence d'objets Computree (items) définis dans PluginShared. Par exemple :

- CT_Scene est une scène contenant le nuage de points complet.
- CT_PointCluster est un groupe de points.
- CT_Circle est un cercle visualisable dans la zone graphique
- CT_Raster2D est une grille pouvant représenter un modèle numérique de terrain

L'atout principal de ce système de modèles et de pouvoir organiser les résultats. Par exemple si l'on souhaite ajouter un cercle à chaque arbre détecté, il suffit de spécifier un modèle d'entrée contenant un CT_PointCluster symbolisant une tranche d'arbre puis de définir en modèle de sortie un groupe contenant un CT_PointCluster et un CT_Circle. Ainsi l'étape va récupérer le résultat d'entrée et pour chaque CT_PointCluster lui associer un CT_Circle. Le résultat est donc ordonné, chaque groupe symbolisant un arbre contiendra maintenant des informations propres à un CT_PointCluster et CT_Circle. La figure 7 illustre la notion d'étape, modèle et résultat appliqué à l'exemple précédent.

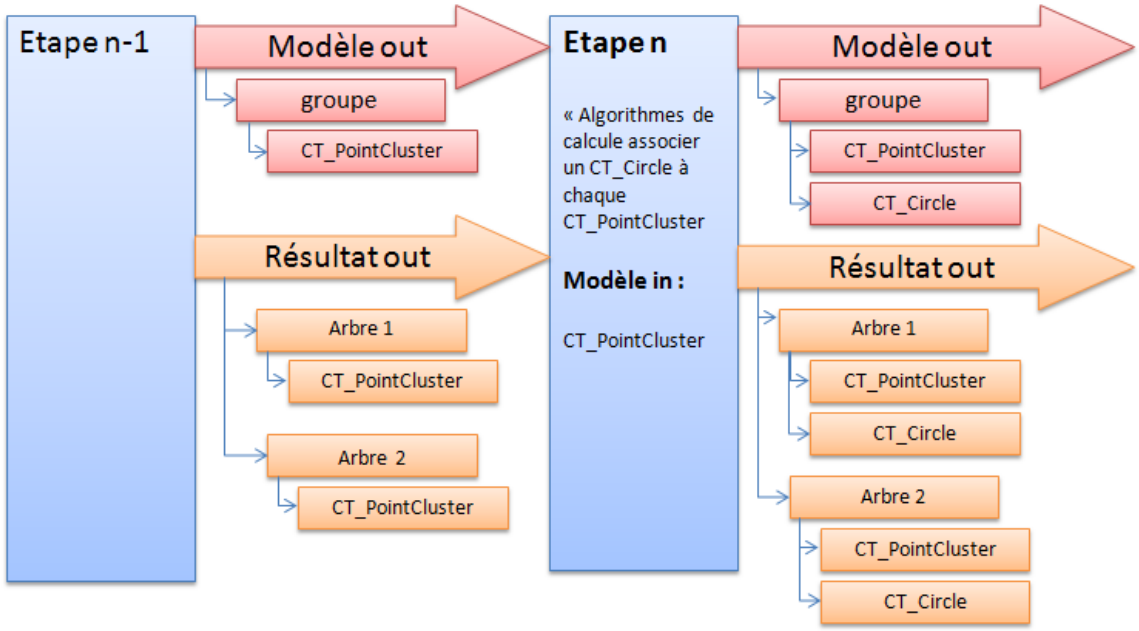


Figure 7 : Schéma de fonctionnement des étapes et des modèles Computree

Les notions d'étapes, de modèle d'entrée, de modèle de sortie, de résultat d'entrée et de résultat de sortie ont été présentées. Computree permet d'accéder à chacun de ces éléments via le gestionnaire d'étape et le gestionnaire de modèles. Le gestionnaire d'étapes permet d'ajouter et de supprimer des étapes, mais également de visualiser l'enchainement des étapes. Pour chaque étape il liste les résultats et permet d'accéder à ses items et aux modèles. Une fois dans le gestionnaire de modèles il est possible de naviguer dans l'arborescence du résultat afin de colorier ou d'afficher les éléments sélectionnés. Cette fonctionnalité est très utile pour visualiser uniquement une partie du résultat. Des fonctionnalités d'ordre ergonomique ont aussi été ajoutées. Par exemple des boutons de gestion de vues ou d'export de nuage de points 3D. Le logiciel offre aussi la possibilité de

synchroniser plusieurs vues entre elles pour comparer deux résultats ayants subi des étapes paramétrées différemment.

En conclusion Computree avec PluginShared offre un panel de fonctionnalités très utiles pour développer rapidement et efficacement un nouveau plugin. Cela présente l'avantage d'éviter aux équipes de recherche souhaitant utiliser Computree de perdre du temps inutilement et de se concentrer sur le développement d'algorithmes de traitement.

5) Les étapes de traitements

L'objectifs de ce paragraphe est de présenter les fonctionnalités de chacune des étapes utilisés durant le stage. Sachant que plusieurs combinaisons d'étapes avec différents paramétrages d'étapes sont possibles, l'ordre dans lequel sont décrites les étapes de traitements n'est qu'une possibilité parmi différentes combinaisons testées durant mon stage.

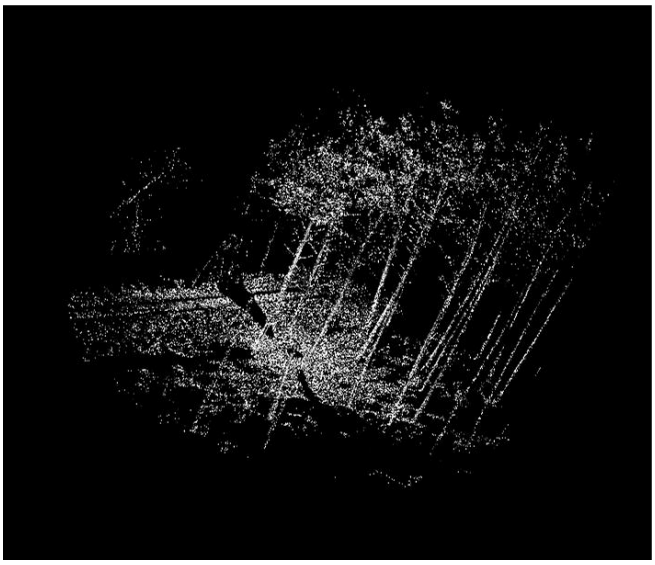


Figure 8 : Etape d'extraction de placette

Dans cette étude deux plugins ont été utilisés :

- le plugin Onf Ensam, qui permet l'import d'un fichier de points 3D, la détection du sol, la détection des arbres, l'ajustement de cylindres sur les points de végétation et finalement une estimation du diamètre à 1m30.
- Le Plugin Onf Lsisv permet de détecter et filtrer les sections d'arbres représentant des fourches ou les sections ne correspondant pas à une portion de cylindre.

Etape d'import

La première étape consiste à importer un fichier contenant un nuage de points et le mettre en mémoire. Les tailles des fichiers représentants des placettes numérisées en mono scan avoisinent 1Go. Il est indispensable de disposer de matériel adapté en termes de capacité mémoire (en particulier pour les multi-scan*). Les formats .xyb, .csv et .ascii sont les trois formats de fichiers gérés par Computree

Etape d'extraction de placette

Cette étape allège la taille du nuage de point. Il a été choisie de ne conserver que les points dans un cylindre d'un rayon de 17 m de l'axe central. Les points avec une altitude aberrante de la plage -50m et +1000m, dégradent considérablement les temps de calcul, ils ont aussi été écarté à cette étape (voir figure 8).

Etape d'extraction de sol

Durant cette étape, un quadrillage en x y divise la placette en plusieurs zones, dans lesquelles le minimum local en z est cherché. L'ensemble des points sélectionnés représentent le MNT* (modèle numérique de terrain). Dans un deuxième temps les points sont lissés et une extrapolation est fait dans les zones occultées, pour obtenir un MNT cohérant. Ensuite tout point au-dessus d'une distance paramétrable du MNT est considéré comme de la végétation, le reste est considéré comme du sol (voir figure 9). Cette étape permet donc de réduire considérablement le nombre de points à traité, en appliquant l'ensemble des autres étapes de traitement sur le nuage de point de la végétation.

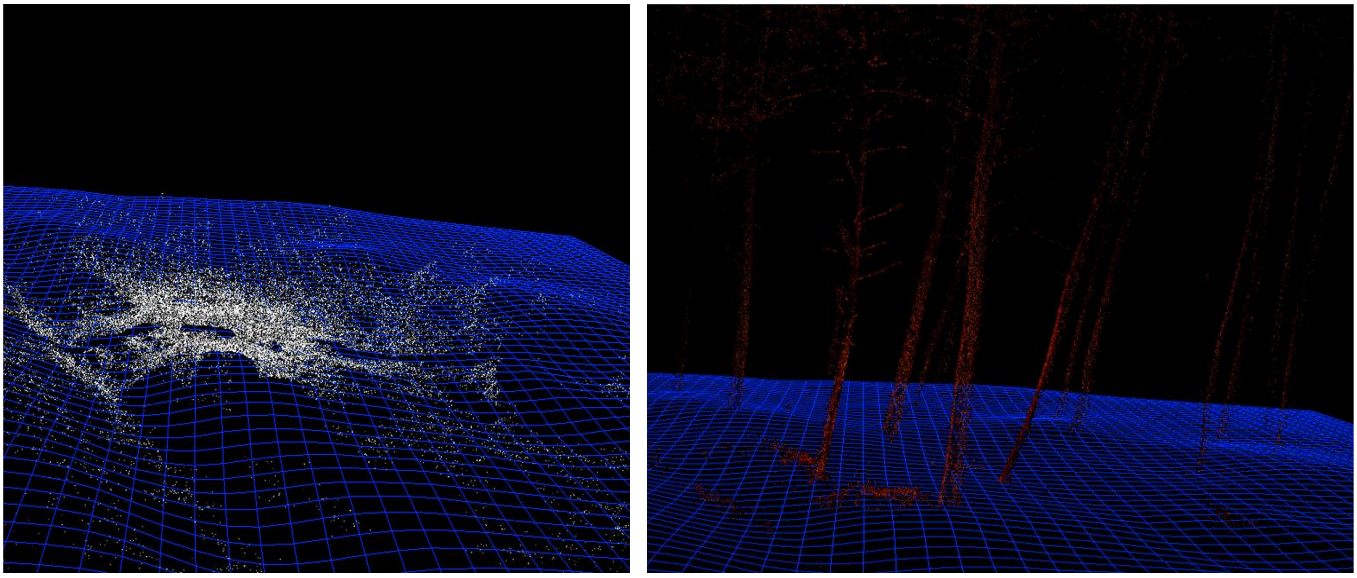


Figure 9 : Dissociation du sol (à gauche) et de la végétation (à droite) par Computree

Etape de clustering

Depuis le nuage de points de la végétation, l'espace est découpé en tranches horizontales espacées de 1 cm. Dans chacune des tranches, les points distants de moins de 3 cm (distance paramétrable) forment des groupes (clusters). A ce stade dans le cas d'un mono scan, les clusters des arbres prennent la forme d'arcs de cercle plus ou moins perturbés par la végétation basse.

Etape de filtrage des cluster

On filtre une première fois les clusters en éliminant tous les groupes de moins de 5 points (valeur paramétrable sachant qu'il faut au moins 3 points, pour ajuster un cercle).

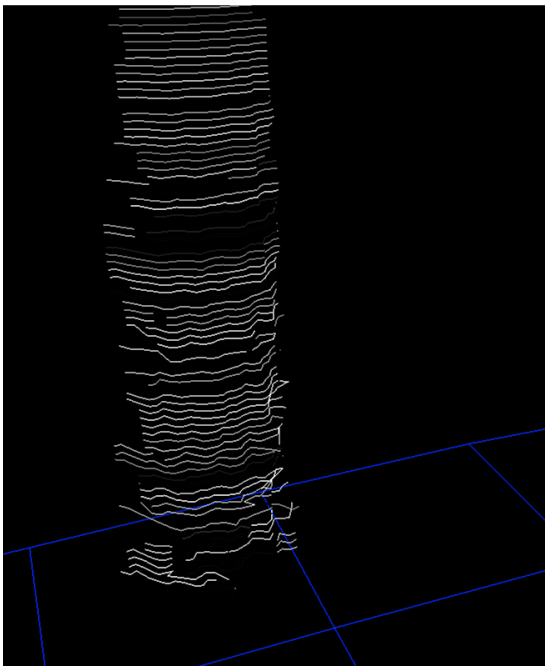


Figure 10 : Polylignes sur Computree

Etape de jeté de particules

Cette étape lance à plusieurs reprises un nombre de particules proportionnelle au périmètre de la boite horizontale englobant de chaque Clusters. Les particules sont aléatoirement lancées, puis chacune des particules est attirée par les points les plus proches. Dans un deuxième temps les particules se repoussent entre elles. Après plusieurs cycles d'attraction répulsion, les particules trouvent une position d'équilibre qui maximise la distance total entre particules. La répartition des particules est homogène et gomme une partie de la rugosité de l'arbre.

Etape de création de polylignes

Cette étape crée une polyligne par groupe de particules sur chaque cluster. Une polyline est une courbe discrète reliant par des segments les points les plus proches entre eux. Cette méthode à l'avantage de fournir la longueur des segments et de leur orientation (voir figure 10).

Etape de filtrage des polylignes

Les polylines sont projetés sur un plan horizontal. Ainsi les clusters représentants des parties d'arbres forment des arcs cercles, alors que les fourches forment deux arcs de cercles et que les autres objets forment un géométrie aléatoire. Le principe de filtrage consiste à utiliser l'espace des tangentes* dans lequel les arcs de cercles sont composé des points alignées. Par conséquent plusieurs segments de Polylignes appartenant au même arc de cercle forment une droite. Cette étape filtre donc les polylines qui représentent mal une droite dans l'espace des tangentes avec les trois critères suivants, erreur max, RMSE et R^2 ajusté.

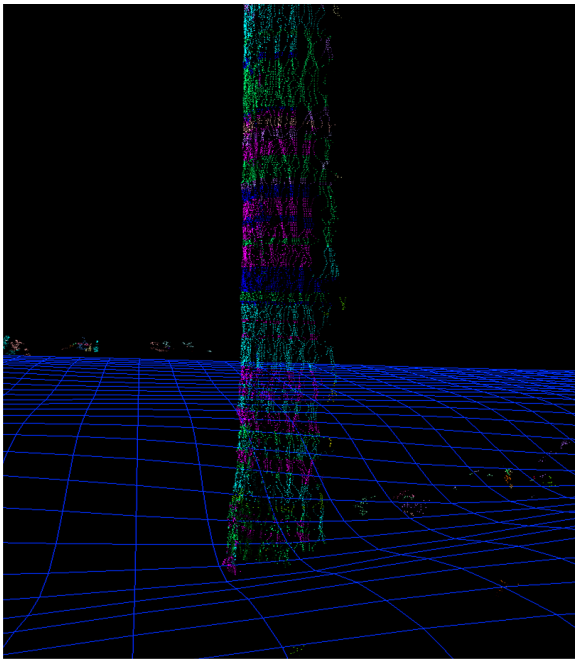


Figure 11 : Création des sections Computree

Etape de détection de section

Cette étape permet de détecter des sections, une sections étant un groupe de clusters proches les uns des autres verticalement (voir figure 11).

Etape de filtrage par nombre de clusters par section

Cette étape permet de filtrer les sections contenant moins de 5 clusters, supprimant ainsi une parties des sections représentant la végétation basse ou le feuillage, qui n'auraient pas été éliminées auparavant.

Etape de fusion des sections voisines

Durant cette étape la taille verticale de clusters est redéfinis, ensuite le squelette composé des barycentres de chacun des clusters est tracé. Finalement les sections ayant des parties de squelette proche horizontalement sont fusionnées (voir figure 12).

Etape de fusion des sections alignées

Durant cette étape la taille verticale de clusters est redéfinis, ensuite le squelette composé des barycentres de chacun des clusters est tracé. Elle associe ensuite les sections dont les extrémités jointives des squelettes sont approximativement alignés .

Etape d'ajustement et de filtrage des cylindres

Cette étape ajuste plusieurs cylindres d'une hauteur paramétrables, aux sections. L'algorithme récupère le squelette engendré par la succession de barycentres des clusters de la section, et pour chaque cluster, il associe le cylindre dirigé suivant le squelette le mieux ajusté suivant le critère des moindres carrées (voir figure13).

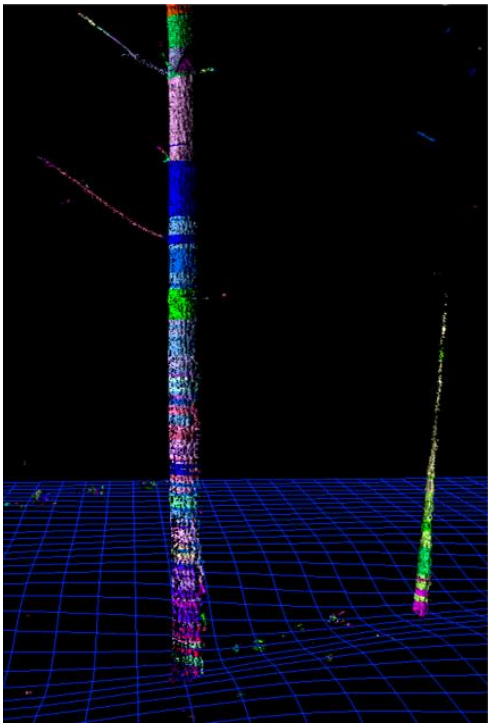


Figure 12 : Fusion des sections voisines

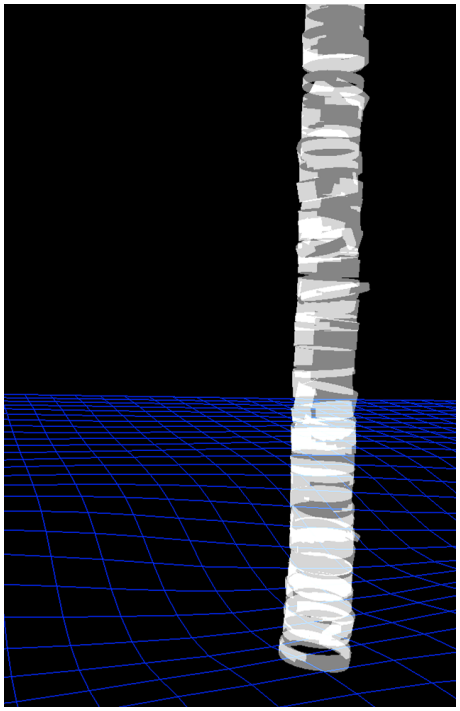


Figure 13 : Etape ajustement des cylindres

Etape de modification des coordonnées en fonction du MNT

Les coordonnées en z sont redéfinies depuis le MNT, afin de positionner les arbres numériques par rapport au sol.

Etape d'extraction des diamètres depuis les cylindres

Les diamètres des différents cylindres d'un arbre numérique sont relevés en fonction de leur hauteur. Les cylindres sont tracé dans un repère diamètre du cylindre, hauteur. Puis une régression linéaire des diamètres en fonction de la hauteur est déterminé afin de lisser les diamètres des cylindres sur +-1 m autour de 1 m 30. Ainsi le diamètre à 1m30 estimé prend en compte la décroissance du diamètre du tronc avec la hauteur en conservant la possibilité de réduire l'effet de certains cylindres mal ajustés. Un point de référence représentant l'intersection entre le squelette de la section et le MNT est aussi déterminé.

Etape de calcul des indicateurs arbres

J'ai développé cette étape pour calculer une série d'indicateurs sur chacun des arbres détectés par Computree. Les indicateurs seront détaillés précisément dans le chapitre 4. Une fois calculés, les résultats sont automatiquement exportés sur un fichier csv.

Etape d'extraction de placette depuis un MNT

Cette étape a été développée au cours du stage afin d'extraire d'un nuage de point, les points entre 0,3 m et 2,3m de hauteur par rapport au MNT. Cela accélère considérablement l'étape de calcul des indicateurs placettes.

Etape de calcul des indicateurs placettes.

J'ai aussi développé cette étape afin de calculer des indicateurs basés sur le résultat de l'étape précédente. Ces indicateurs sont automatiquement exportés vers un fichier Excel. Ces indicateurs placettes seront aussi détaillées dans le chapitre 4.

A l'attention des lecteurs désirant plus d'informations sur les étapes de traitement. Computree étant un projet libre, il est possible de contacter Alexandre Piboule du pole R&D de Nancy pour avoir accès au code des différentes étapes.

Lors de cette partie, les fonctionnalités, la structure et le fonctionnement global de Computree furent largement décrits, cela permet de commencer la présentation du jeu de données utilisé pour l'évaluation des performances du logiciel.

III. Jeu de données et critères descriptifs

Le jeu de donnée mis à disposition pour l'étude d'évaluation des performances de Computree a été fourni par l'IGN. Il comporte 370 scans provenant de la campagne de numérisation lidar, des fiches terrains associées ainsi que les données caractéristiques des placettes et des arbres inventoriés correspondants. La numérisation de placettes étant une opération récente pour les équipes de l'IGN, une grande partie de pré-traitement a dû être effectué durant le stage. Cela a conduit la rédaction d'un rapport présentant l'ensembles des difficultés rencontrées durant la phase de mise en forme des données, et les recommandations pour éviter certains problèmes à l'avenir. Ce document est disponible en annexe. Afin de présenter en détail le jeu de données, une première partie est consacrée à la description des protocoles utilisés, puis deux parties détaillent les critères descriptifs des placettes et des arbres. Finalement une dernière partie présente de manière statistique le jeu de données.

1)Protocoles IGN

a)Protocole d'inventaire forestier

A l'issue d'une étape de photo-interprétation, environ 8000 placettes par an sont choisies de manière aléatoire. Elles sont ensuite visitées par une équipe de terrain de l'IGN effectuant un certain nombre de mesures dans un rayon de 25 m, à parti du centre de la placette. Un relevé floristique, une description du sol, ainsi qu'une description du peuplement sont réalisés (la figure suivante présente dans quelle zone sont effectués ces relevés). Concernant la mise en place du jeu de données, les arbres sont recensés et mesurés dans un rayon de 15 m autour d'un piquet repère (voir figure 14).

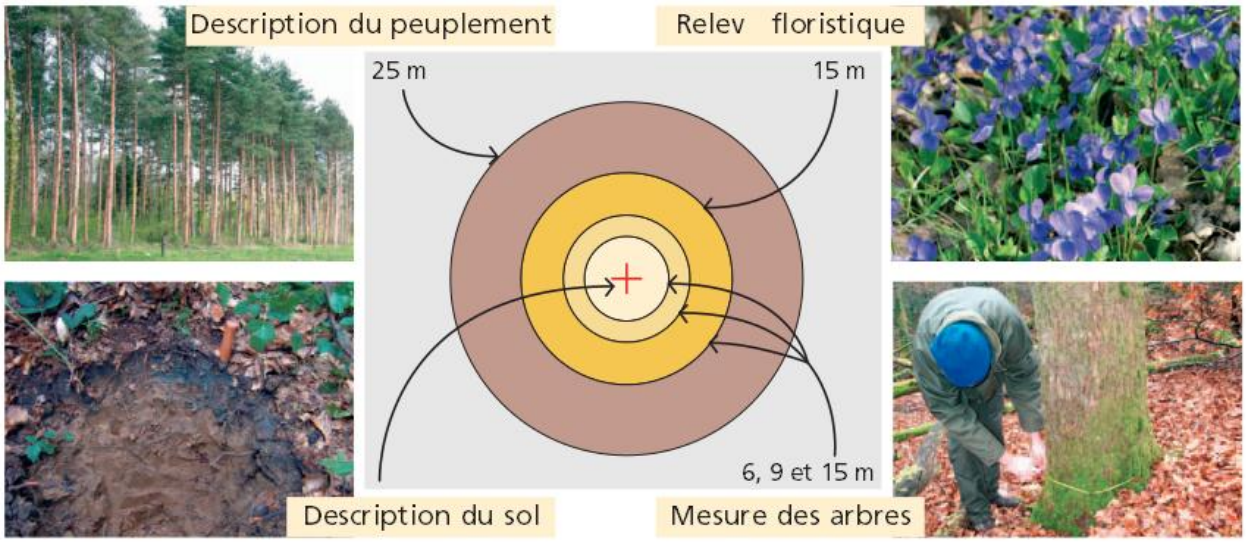


Figure 14 : Schéma des types de relevé par cercles

Afin de simplifier la tache d'inventaire, des arbres considérés comme similaires sont simplifiés. L'espèce et la catégorie du diamètre sont les deux principaux facteurs déterminant quels arbres doivent être simplifiés ou inventoriés. Dans le cas d'un arbre simplifié, uniquement la mesure de diamètre et de hauteur sont réalisés. Les mesures distance et d'azimut ne sont alors pas réalisés. La simplification rend donc impossible une association directe entre les arbres détectés par Computree et les arbres inventoriés. Pour cette étude il a donc été choisi de ne traiter que les arbres non simplifiés. La catégorie de diamètre, elle, définit une zone de recherche où les arbres doivent être inventoriés. Le tableau de la figure 15 résume la distance à laquelle les arbres sont inventoriés en fonction de leur catégorie de diamètre.

	Catégorie de diamètre en cm	Distance au piquet repère en m
Petit bois	[7.5 , 22,5[	6
Moyen bois	[22 .5 ,37.5[	9
Grand bois	≥37,4	15

Figure 15 : Tableau de rayon d'inventaire en fonction de la catégorie de diamètres

Concernant les outils de mesure utilisés, la circonférence est relevée au mètre ruban à une hauteur de 1,30m. La hauteur 1,30m est mesurée en déroulant le ruban le long de la tige, de manière à atteindre la position la plus haute depuis le pied de l'arbre. La position est relevée au télémètre[1] depuis le piquet repère en visant le centre de l'axe de l'arbre 30cm au dessus de son pied.

b) Protocole de numérisation de la placette

En mono-scan le lidar terrestre est positionné au dessus du piquet repère en pointant une direction sans arbre afin d'éviter des discontinuités entre le début et la fin de scan. Suivant les équipes la végétation basse a été dégagée ou non. L'azimute de départ est relevé, l'appareil ne disposant pas de visée, le relevé de l'azimut s'effectue avec une incertitude de +-4GRADs, ce qui est pénalisant pour l'association entre les arbres numériques identifiés par Computree et les arbres inventoriés. Le modèle utilisé par l'IGN durant ses campagnes de mesure est le Faro Photon tm 80. Ses caractéristiques principales sont résumées sur la figure 16. Les résolutions verticales et horizontales de scan sont respectivement 0,037422 et 0,0359879 RAD. Pour donner un ordre de grandeur un objet situé à 15 m est parcouru avec un pas de 9mm vertical et de 8mm horizontal.

Figure 16 : Caractéristique du lidar utilisé

[1] Vertex : appareil de mesure de distance utilisé en foresterie.

2) Critères placettes

Une partie des données ont été fournies directement par l'IGN avec des critères propres à chacun des arbres et des placettes. Une autre partie fut extraite des fiches terrains ou calculée à partir d'autres données. Cette sous partie a donc pour objectif de présenter de manière exhaustive l'ensembles des critères descriptifs utilisés dans l'étude.

Pour la grande majorité des placettes scannées au LiDAR Terrestre, une fiche terrain est associée. Les données relevées sur les fiches terrains dépendent des années de campagne et des zones où ont été effectués les scans. En dépouillant manuellement chacune des fiches terrains les informations suivantes ont pu être récupérées :

- Un numéro de placette.
- L'année de la campagne de mesure.
- La zone de la mesure (Bordeaux/Caen/Lyon/Montpellier/Nancy).
- L'orientation de départ du scan central en grade.
- Les remarques (notés quand pertinentes pour l'étude).

Un certains nombre d'informations récupérées mais pas notées de manière systématique ont aussi été relevées (Indication sur la pente ,présence de limites, Indication sur le comportement vis-à-vis de la vibration du LiDAR durant la mesure, Indication sur la prise de scan en couleur, Indication sur l'intensité du vent, type de peuplement, Indication sur la météo, indication sur le sol. Les données ainsi relevées depuis les fiches terrains sont pour la plus part difficilement exploitables. Cela aurai pu être intéressant notamment pour le critère de vent. Les données d'inventaires fournies par l'IGN sous la forme d'un tableau contiennent les informations suivantes :

- L'identifiant de la placette.
- Un identifiant de campagne de mesure.
- Le département dans lequel a été fait la mesure.
- Le type de propriété (0 hors territoire, 1 domanial , 2 communal, 4 privé, 8 occulte, NA non renseigné) notée pro.
- La structure forestière notée SFO (0 pas de structure : situation ou la structure forestière n'est pas définie/ 1 futaie régulière : régime de futaie avec une distribution verticale régulière/ 2 futaie irrégulière : régime de futaie avec une distribution verticale irrégulière/ 3 mélange de futaie et taillis : régime de mélange de futaie et taillis/ 4 taillis : régime de taillis.)

En partant des données acquises, le calcul d'autres critères descriptifs a été automatisé grâce à des macros Vba :

- La surface terrière des petits/moyens/gros bois des arbres simplifiés et non simplifiés par placette. Les petits bois ont un diamètre compris entre 7,5 cm et 22,5 cm, les moyens entre 22,5 cm et 37,4 cm et les gros au-dessus de 37,4 cm. La surface terrière totale est pondérée en fonction de la taille de la zone d'inventaire de chacune des classes de diamètres des arbres.
- La surface terrière totale en sommant les trois surfaces terrières précédentes.
- Le pourcentage de tiges de petit/moyen/gros bois.
- La distinction entre un peuplement de feuillus/résineux/mixte en utilisant les espèces des arbres inventoriés sur la placette.

Lors de la conversion des fichiers j'ai pu visualiser l'ensemble des scans, et relever les informations suivantes :

- Distinction entre un peuplement en feuilles ou hors feuille (sachant que les peuplements de résineux ont été considérés comme hors feuille, sauf dans le cas de la présence d'un sous étage de feuillus en feuille).
- Remarques sur les placettes représentant des situations exceptionnelles.

3) Les critères arbres

Les données d'inventaires fournies par l'IGN sous la forme d'un tableau contiennent les informations suivantes :

- a : numéro de l'arbre recensé sur la placette.
- Espar : nom de l'espèce.
- Veget : état de végétation de l'arbre observé(0/5/C/A). Arbre vivant sur pied -> 0. Arbre mort sur pied -> 5. Arbre mort sur pied cassé -> C. Arbre chablis -> A.
- Dpr : distance de l'arbre au piquet repère (centre de la section de l'arbre à 30 cm du sol) en m.
- Azpr : azimut de l'arbre depuis le piquet repère (centre de la section de l'arbre à 30 cm du sol), indiqué en radians.
- C13 : circonférence à 1,30 en m.
- Htot : hauteur totale en décimètres.

Les informations extraites des fiches terrains sont :

- Le numéro des arbres coupés.

Les critères calculés depuis les données d'inventaires de l'IGN sont :

- La surface de l'arbre à 1 m30 noté s13 calculé avec le c13
- Un identifiant comportant l'identifiant de la placette et le numéro de l'arbre afin de disposer d'un identifiant unique par arbre.

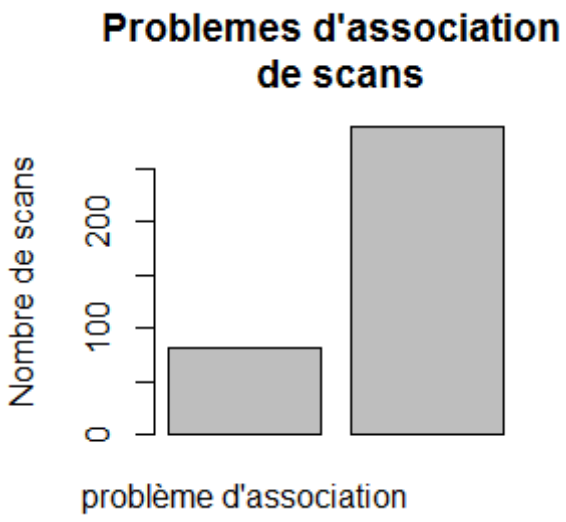


Figure 17 : Histogramme problèmes d'associations de scans

4)Jeu de donnée

Dans cette partie, les résultats de une analyse descriptive du jeu de données sont présentés. Durant le regroupement des informations plusieurs problèmes d'associations, expliqué en annexe I, ont réduit le nombre de placettes exploitables à 292 (voir figure 17).

L'IGN a organisé le territoire français en 5 grandes zones, la figure 18 montre que le nombre de scans associés sont répartis équitablement entre ces différentes zones et donne un aperçu de l'intensité des campagnes 2010, 2011, 2012. Cependant la carte représentant le nombre de placettes scannées par département (figure 19) montre que le jeu de donnée n'est pas encore suffisant pour être considéré comme représentatif de l'ensemble du territoire français, ce qui ne pose pas de problème pour cette étude.

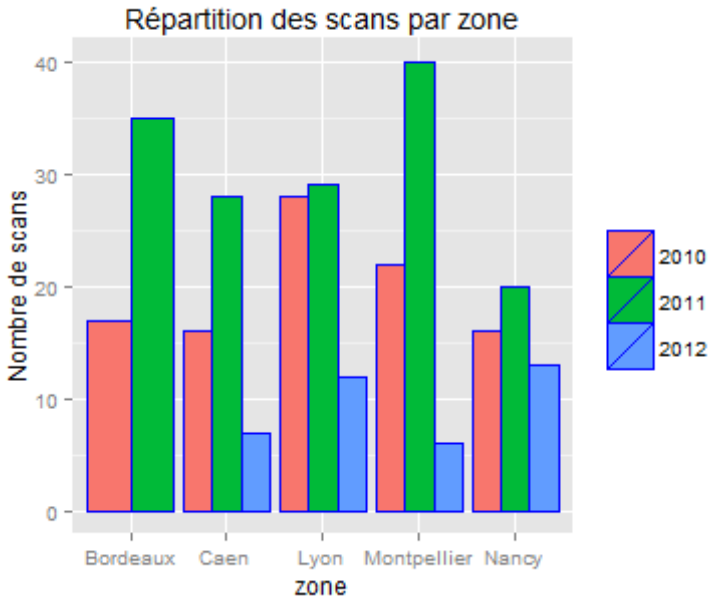


Figure 18 : Histogramme du nombre de scans par année et par zone

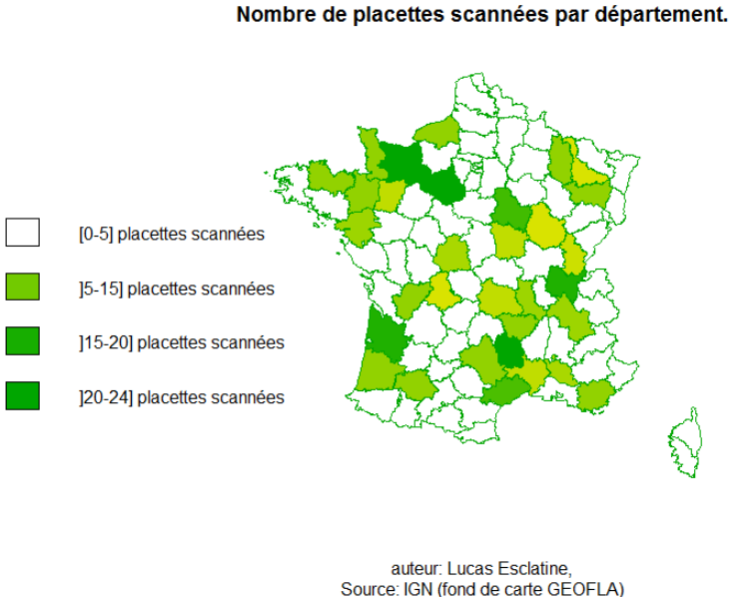


Figure 19 : Carte du nombre de scans par département

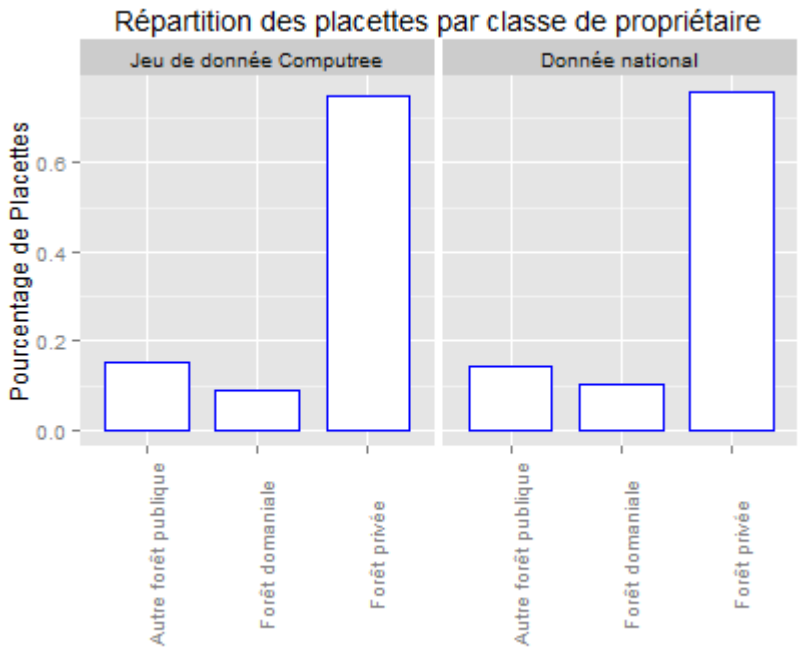


Figure 20 : Histogramme nombre de scans par classe de propriétaire

La figure 20 représante la répartition des placettes suivant la classe de propriétaire, montre qu'il y a une très bonne correspondance en proportion entre le jeu de donnée de l'étude et les données nationales (statistiques issue du rapport de l'IFN ''Chiffre clès forêt 2012 '').

La figure 21 donne la répartition du nombre de placette en fonction de la structure forestière, ainsi il semblerait que le nombre de placette de structures, forêt ouverte, futaie irrégulière incitent à la prudence lors de l'interprétation des résultats selon ses critères. A l'inverse une comparaison futaie régulière, taillis sous futaie, taillis est envisageable.

La figure 22 donne la répartition des placettes en feuilles ou hors feuille selon le type d'essences inventoriées présentent sur la placette. La répartition entre les peuplements en feuilles et hors feuilles est suffisante pour tenir compte de ce critère, de même pour le critère de type d'essence. La figure 23 quant à elle donne la proportion d'arbre inventorié par essence.

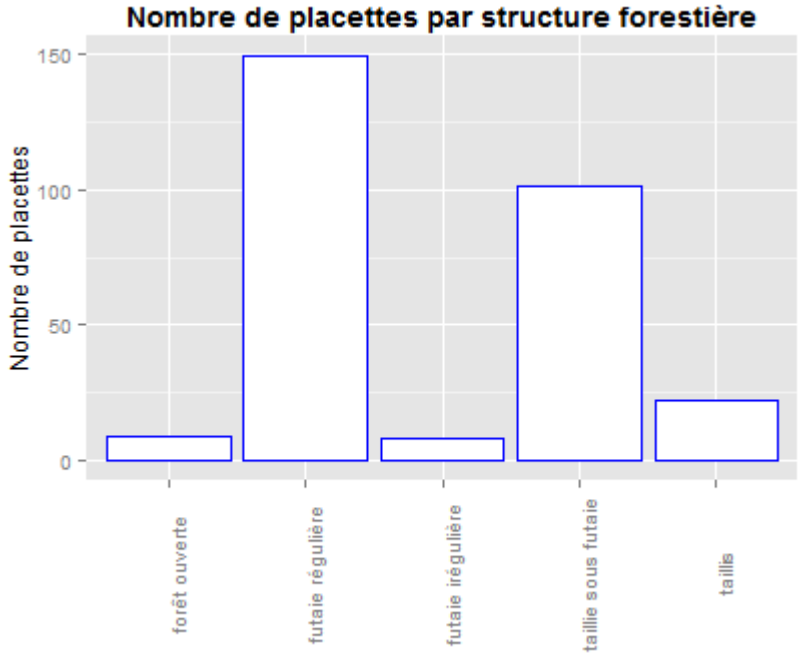


Figure 21 : Histogramme du nombre de placettes par structure forestière

Bien que la campagne de numérisation lidar n'est qu'à son début, le jeu de données suffit déjà pour évaluer l'influence des critères de classe de propriétaire, structure forestière et de type de feuilles. A ce stade le jeu de données n'est pas encore suffisant pour être représentatif. Cependant le nombre de placettes offre une variabilité correcte pour l'évaluation des performances de Computree.

Au cours de cette partie, le jeu de données utilisé pour l'évaluation des performances de Computree fut brièvement présenté. Il est temps

d’exposer les résultats de l’étude.

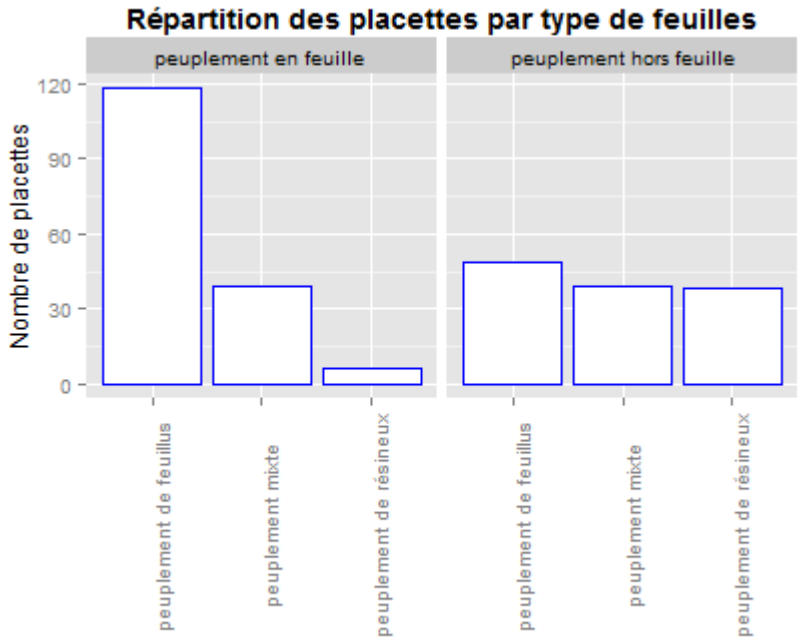


Figure 22 : Histogramme de la répartition des placettes par type de feuille

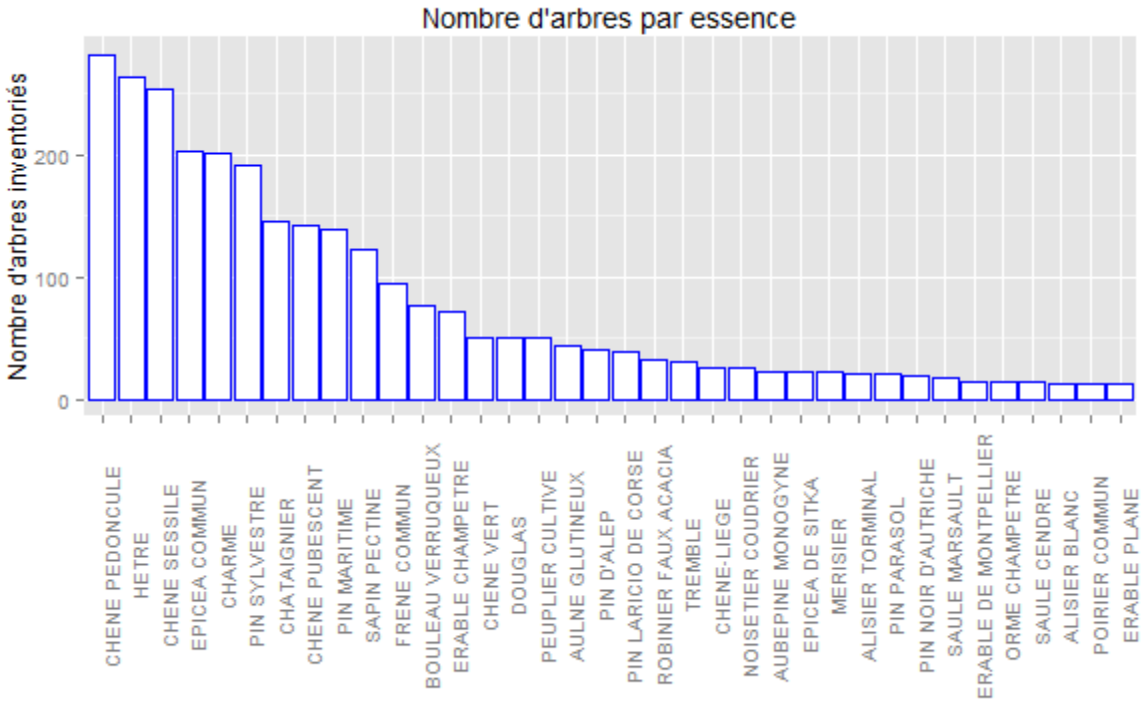


Figure 23 : Histogramme du nombre d'arbre inventoriés par essence

IV. Evaluations des Performances Computree

1)Indicateurs explicatifs

a) Introduction

Après avoir présenté rapidement le jeu de données utilisé, dans un premier temps les indicateurs mis en place pour évaluer et expliquer les performances de computree sont présentés, puis le protocole utilisé est décrit, et finalement les résultats obtenus sont exposés et commentés.

L'un des objectif de Computree été de chercher des indicateurs permettant de détecter les cas ou Computree fonction potentiellement mal.

Le pourcentage de volume occulté par la végétation basse et le nombre de points par placette sont tout les deux des indicateurs de niveau placette cherchant à représenter l'intensité de l'occultation global de la placette. Il permet de signaler des situations comme une placette avec une végétation basse très dense, ou plusieurs arbres proche du lidar masquant une grande partie de la placette.

A l'échelle d'un arbre l' indicateur de densité de point par arbre, le nombre de points par arbre, le nombre de cylindre, et le nombre de cluster ont pour objectif d'évaluer la quantité d'information géométrique minimum nécessaire à une bonne estimation du diamètre de l'arbre. L'indicateur de forme de squelette de l'arbre numérique évalue l'impact de la géométrie du squelette des arbres numériques sur l'estimation du diamètre, et finalement les indicateurs de répartition des arcs de cercles vérifient l'impacte de l'homogénéité de la répartition des points en hauteur, toujours sur l'estimation du diamètre à 1 m 30.

b) Indicateurs placettes

Le pourcentage de volume occulté par la végétation basse.

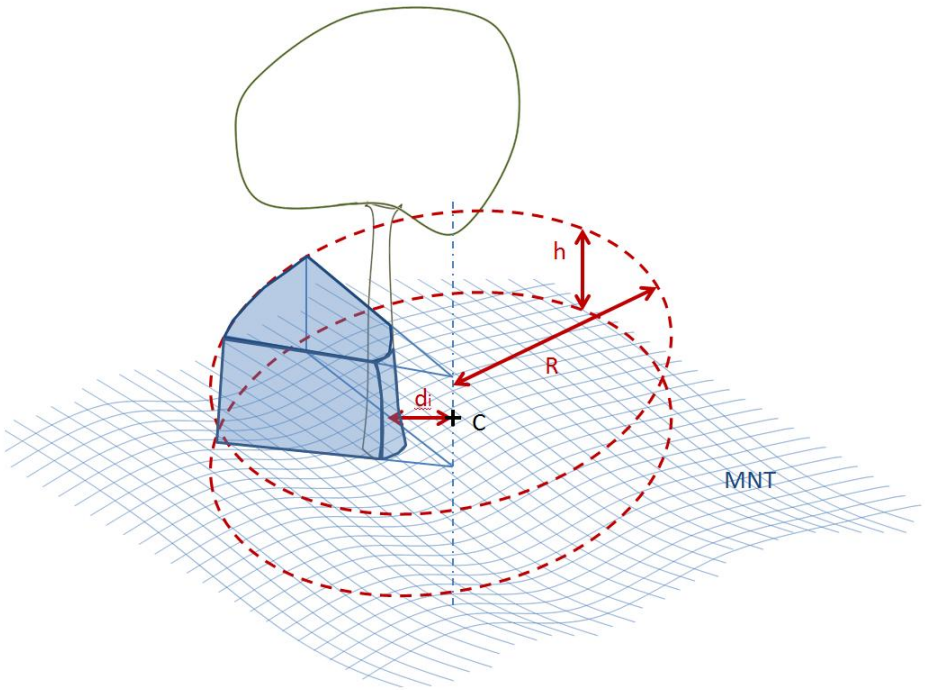


Figure 24 : Schéma de calcule de l'indicateur de pourcentage de volume occulté

Cet indicateur est calculé à l'aide de Computree lors de l'étape de calcul des indicateurs placette à la suite de l'étape d'extraction du sol par rapport au MNT. Ainsi le modèle en entrée d'étape est composé du MNT de la placette et de l'ensemble des points compris entre 0,30 m et 2,3 m au dessus du MNT. L'algorithme de calcul développé est reporté en annexe II. Le pourcentage de volume occulté par la végétation basse représente le volume occulté sur le volume totale entre le MNT et le MNT décalé vers le haut d'une hauteur h dans un rayon R. Le schéma de la figure 24 présente les paramètres de calcul de cet indicateur. La frontière de l'espace où est calculé le MNT est symbolisé par les

deux cercles rouges, la distance entre le centre de la placette, C et un point appartenant à la végétation est noté di. Dans cet exemple le pourcentage de volume occulté est le volume occulté par le tronc de l'arbre en bleu divisé par le volume entre les deux cercles rouge.

Pour calculer le volume occulté d'un point, il est possible de faire l'intégrale de portion de sphère infinitésimal depuis le point jusqu'à la frontière. Or à une distance r une portion de sphère peut être approximé par un rectangle d'une aire de $r^2 * \Delta\theta * \Delta\alpha$ ($\Delta\theta * \Delta\alpha$ la résolution du LiDAR sur les deux angles de rotation). La figure 25 résume le calcul de volume infinitésimal.

Ainsi en intégrant le volume infinitésimal du point de la végétation jusqu'à la frontière on obtient le volume occulté par le point, calculable avec la formule suivante.

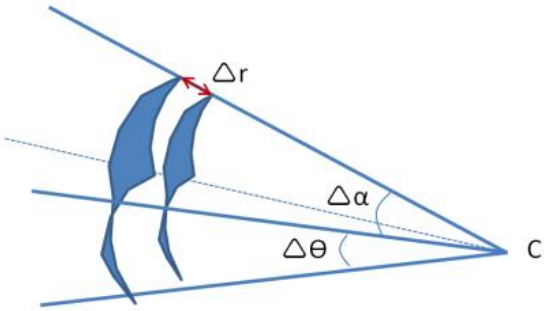


Figure 25 : Schéma explicatif du calcul de volume occulté

$$\Delta V = \int_d^R r^2 . dr * \Delta\theta * \Delta\alpha$$

$$\Delta V = \frac{\Delta\theta * \Delta\alpha}{3} * (R^3 - d^3)$$

Le pourcentage de volume occulté par la végétation basse se calcule alors en faisant la somme des volumes cachés par chacun des points (la formule est donnée ci-dessous).

$$I_{Vocculté\ vegt} = \frac{\Delta\theta * \Delta\alpha}{4} * \frac{\sum_{i \in véget\ basse}(R_i - d_i^2)}{V_{tot}}$$

Avec $V_{tot} = h * \pi * 15^2$

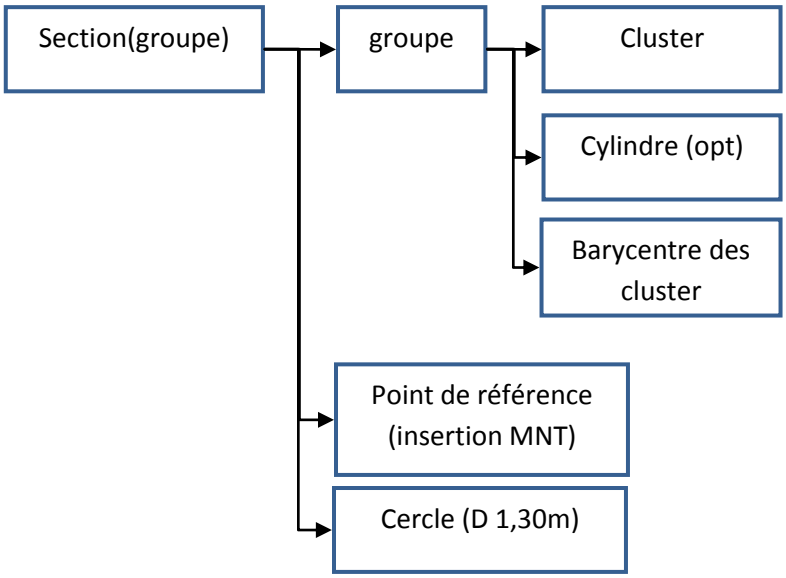


Figure 26 : Schéma du model IN de l'étape de calcul des indicateurs

c) Indicateurs arbres

L' indicateur de densité de point par arbre, l'indicateur de forme de squelette de l'arbre numérique et les indicateurs de répartition des arcs de cercles sont tous calculés lors d'une même étape sur Computree à la suite de l'enchainement d'étapes de traitement de données. L'algorithme de calcul développé est reporté en annexe III. Le modèle d'entrée de l'étape de calcul des indicateurs arbres est présenté sur la figure 26. Ainsi pour chaque section représentant un arbre numérique, est associé un cercle représentant le diamètre de l'arbre à 1m30, un point de référence représentant le centre géométrique du pied de l'arbre au niveau du sol et plusieurs groupes contenant chacun un groupe de points, le cylindre correspondant ajusté sur l'axe du squelette de l'arbre, et un point de

référence symbolisant le barycentre du groupe de points (voir figure 26 et 27).

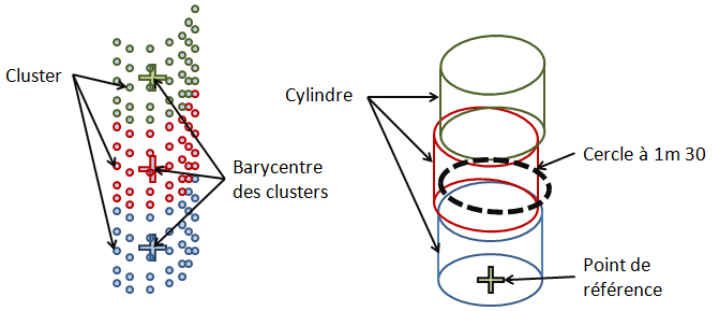


Figure 27 : schéma de la structure du modèle de calcule des indicateurs

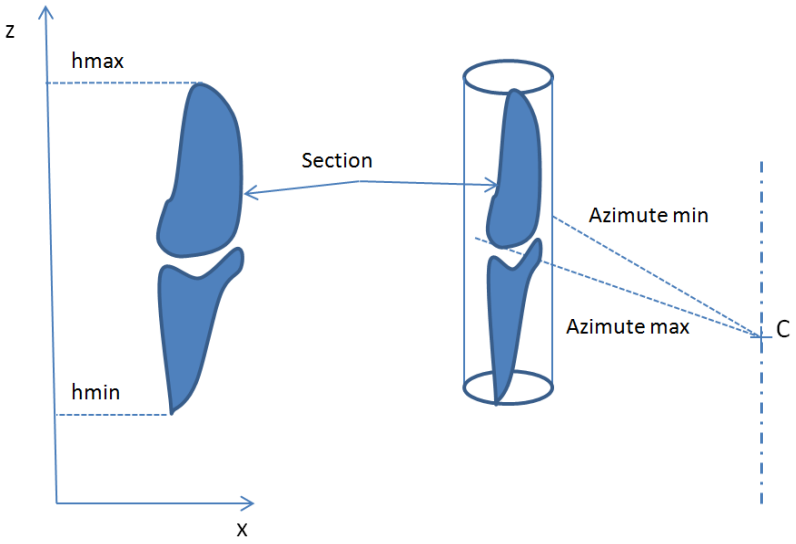


Figure 28 : Schéma des paramètres de l'indicateur de densité de points par arbre

Indicateur de densité de points par arbre

Description

L'indicateur de densité de point par arbre est le rapport entre le nombre de points de la section de l'arbre numérique divisé par le nombre de points maximum que cet arbre pourrait avoir, pour la tranche d'une hauteur comprise entre 30 cm et 2 m30. Cet indicateur estime si la section décrivant l'arbre est constituée d'un nombre de points suffisant. L'algorithme de calcul compte alors le nombre de points de la section, d'une hauteur comprise dans le bon intervalle, puis il cherche le point le plus haut (noté hmax), le point le plus bas (noté hmin), le point avec l'azimut maximum et minimum (notés azimut max et azimut min voir figure 28).

Afin dévaluer le nombre de points maximum que la section pourrait avoir au maximum, le nombre de lignes de scan horizontales et verticales balayées par le lidar sont calculées d'après les formules suivantes (voir figure 29).

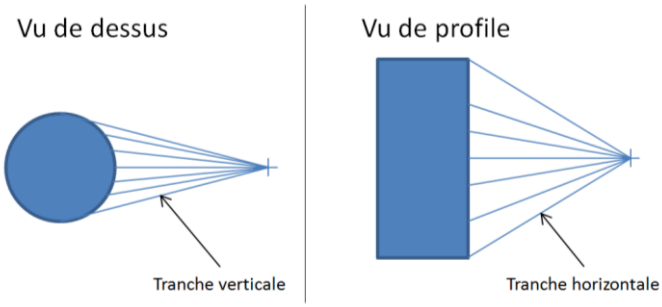


Figure 29 : Schéma du calcul du nombre de tranches horizontales et verticales

$Nb\ Tranche\ hor = \frac{(h_{max} - h_{min})}{\Delta\theta * d}$ avec d (la distance du point de référence de l'arbre au centre)

$Nb\ Tranche\ hor = \frac{(azimut_{max} - azimut_{min})}{\Delta\alpha}$ avec (

$respectivement\ \Delta\theta\ et\ \Delta\alpha\ la\ résolution\ zénital\ et\ azim$

Indice de forme de squelette de l'arbre numérique

Cet indice est représentatif de la courbure du squelette des arbres numériques. Le squelette est formé par la succession des barycentres des clusters (d'une épaisseur de 10cm dans l'étude). Ainsi un arbre avec une courbure simple et avec un squelette représentatif de la courbure, a une indice de forme nul. Il se calcule en faisant la somme des angles entre les différents points du squelette moins l'angle total. Ainsi un arbre avec une courbure homogène aura un indice nul et à l'inverse un arbre avec une forme de tige plus chaotique aura un indice de forme de squelette très grand (voir figure 30).

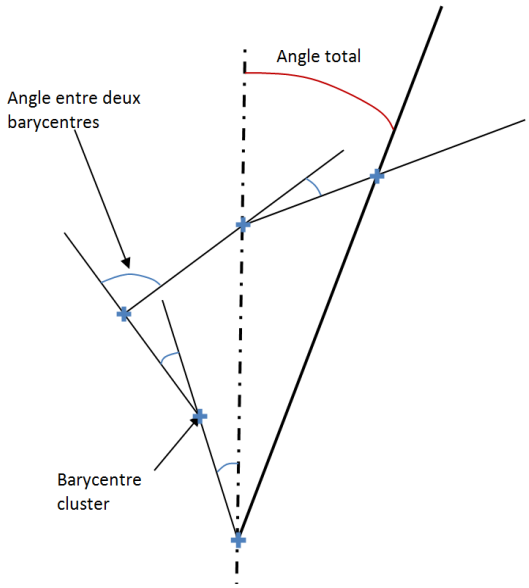


Figure 30 : Schéma de l'indice de forme de squelette

Indicateur de répartition des arcs de cercles

Une section représentant un arbre numérique est découpée en tranches d'une hauteur paramétrable (10 cm). Chaque tranche représente un arc de cercle si elle contient un nombre de points supérieur à 5. L'indicateur de répartition des arcs de cercles permet de connaitre si les arcs de cercles servant à construire les sections sont répartis de manière homogène sur la hauteur ou non. . Cet indicateur se décompose en trois sous indicateur :

- Le taux d'arcs de cercles présents dans la section, calculé avec la formule suivante. Il représente le rapport entre le nombre d'arcs de cercles d'une section et le maximum d'arcs de cercles qu'elle pourrait avoir.

  $I_{taux\ d'arcde\ cercles} = \frac{Nombre\ d'arc\ de\ cercle\ de\ la\ section}{H_{section}/\Delta h}$ avec Δh la hauteur des clusters.

- La hauteur de l'arc de cercle médian sur la hauteur total.
- Le rapport de la hauteur du quartile inférieur sur hauteur théorique du quartile inférieur si la répartition des arcs de cercles est homogène, calculé avec la formule suivante

$$\frac{Hauteur_{quartile\ inf}}{\frac{(hmin + (hmax - hmin)}{4}}$$

- Le rapport de la hauteur du quartile supérieur sur hauteur théorique du quartile supérieur si la répartition des arcs de cercles est homogène, calculé avec la formule suivante

$$\frac{Hauteur_{quartile\ sup}}{\frac{(hmin + (hmax - hmin)}{4}}$$

2)Protocole d'évaluation de Computree

Le système de modèle d'entrée et de sortie permet une grande souplesse dans l'enchainement des étapes de traitement. Par exemple un même filtre peut être appliqué à deux endroits différents ou même deux fois d'affilés. Cela est important car les performances de Computree sont dépendantes des étapes de traitement utilisées. Après une première phase d'optimisation des paramètres d'étapes sur un jeu de 5 placettes avec des caractéristiques bien distinctes. Un ensemble de 292 placettes a été traité automatiquement grâce au mode Batch de Computree. La combinaison d'étapes ainsi choisi lors de l'étude est résumé dans le tableau suivant (figure 31).

Nom étape	Paramètres	Valeurs paramètres	Commentaire
Import du fichier			Import de fichier .xyb
Extract Plot	R placette Z min Z max	17 m -50 m 1000 m	Sélection des points à moins de 17 m de l'axe centrale de la placette. Une limite en Zmin et Zmax est la pour filtrer les points aberrants .
Extract Soil 03	Résolution de la grille Epaisseur sol Densité Minimum sol Distance points isolés Distance lissage Interpolation Lissage	50 cm 32 cm 200 pts/m 3 2 cases Oui Oui	Calcul du MNT et détection du nuage de point appartenant à la végétation.
Extract Plot Based on MNT	Z min Z max	0 m 4 m	Garde les points entre le sol et 4 m au dessus, pour accélérer les étapes suivantes.
Horizontal Clustering 04	Distance point/groupe Epaisseur tranche	3 cm 1 cm	Formation des clusters, pour chaque tranche horizontale.
Filter Cluster By Size	Nombre point min	5 pts	Filtre les clusters de moins de 5 points.
Throw Particules 05	Distance moyenne entre particules Nb voisins Rayon de répulsion	2 cm 1 1,8 cm	Jeté de particule sur les clusters afin d'homogénéiser le nombre de particules par cluster.
Creat Polylines 02			Tracé des polylignes sur les particules.
Filter Arc Polylines	RMSE Rayon arc max	50 ° 200 cm	Les polylignes représentant mal un arc de cercle sont filtrées.
Detect Section 06	Distance en z	+/- 20 cm	Fusion des clusters proches entre eux verticalement dans des sections.
Filter Groupe By Groups Number	Nb de clusters min par section	5	Filtre les sections avec moins de 5 clusters.
Merge Neighbour Section 04	Epaisseur cluster en Z Distance de recherche Distance max entre deux ref points à fusionner Delta Z barycentres	10 cm 10 cm 50 cm 20 cm	Fusion des sections proches entre elles horizontalement.
Merge End to End Sections 04	Epaisseur cluster en Z Distance max entre	10 cm 1 m	Fusion des sections dont le squelette des barycentres des clusters sont alignés.

	deux sections Nb barycentres considérés Chevauchement toléré	10 20 cm	
Set Foot Coordinates Vertically			Changement de repère, la hauteur est exprimé par rapport au MNT
Fit and Filter Cylinders in Sections	R min R max Erreur max Erreur max / diamètre Angle max de l'axe	2 cm 80 cm 3 cm 30 % 60°	Ajustement des cylindres sur chaque cluster de chaque section, puis filtrages des cylindres (verticalité, erreur d'ajustement).
Extract Diameters From Cylinders	Hauteur de référence Hauteur min évaluation Hauteur max évaluation Nombre cylindres min	1,30 m 0,3 m 2,3 m 4	Le diamètre à 1m30 est estimé depuis les diamètres des différents cylindres.
Load Field Inventory			Un inventaire avec les données terrain est chargé.
Associate Inventory with Section			Les sections détectées sont associées à l'inventaire précédemment chargé. Le critère d'association est la distance entre le point de référence de la section et la position de l'arbre à 30 cm relevées sur le terrain.
Export Inventory			Exportation du nouvel inventaire
Calculate Tree Indicator			Calcul des indicateurs arbres et export sur un fichier csv.
Calculate Plot Indicator			Calcul des indicateurs placettes et export sur un fichier csv.

Figure 31 : Tableau des étapes de traitement Computree

L'ensemble des paramètres d'étapes non cités dans le tableau ci-dessus ont été utilisés avec des valeurs par default.

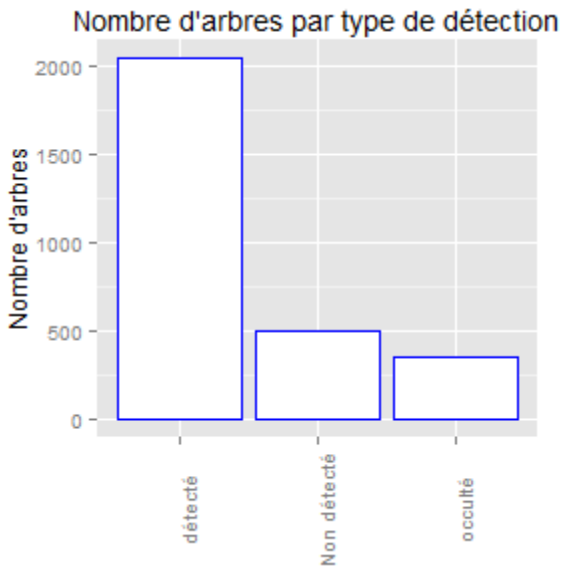


Figure 32 : Histogramme du nombre d'arbres par type de détection

A la suite du traitement des placettes, Computree fournit un nouvel inventaire, associant toutes les caractéristiques des arbres numériques détectés aux arbres inventoriés correspondants. Cependant il a été nécessaire de réassocier une grande partie des arbres manuellement pour corriger les erreurs d'association et de distinguer les arbres occultés des arbres non détectés. En effet il est important de faire la distinction entre un arbre non détecté et occulté, afin de dissocier les phénomènes d'occultation de la capacité de Computree à détecter un arbre.

Lors de la phase de réassociation des arbres numériques, chacun des 2875 arbres inventoriés a été classé manuellement par type de détection. L'histogramme de la figure 32 donnes la quantité d'individu par type de détection.

Les arbres occultés sont donc des arbres avec une occultation de la surface de l'arbre de plus de 80% environ. Les arbres non détectés sont des arbres visualisables sur le scan avec suffisamment d'informations géométriques pour qu'un opérateur puisse les détecter, mais qui n'a pas été détecté par Computree. La proportion d'arbres occultés est liée directement à l'abondance de la végétation basse de la placette, il s'agit d'une limite physique qui peut être réduite en augmentant le nombre de scans.

A la suite de cette réassociation, toutes les données caractérisant les arbres et placettes ont été importé sur R* afin d'analyser statistiquement les résultats, il s'agit de la partie suivante.

Dans un dernier temps, certaines placettes spécifiques ont été visualisé sur Computree afin de comprendre et d'analyser les dysfonctionnements des algorithmes de Computree. Afin de classer les types d'erreurs rencontrées et de proposer des pistes d'améliorations.

3)Résultats

a)Analyse statistique des résultats

L'ensemble des résultats de l'analyse statistique a été obtenu en croisant les données de description des placettes et des arbres avec l'inventaire fourni par Computree et avec les données indicateurs placettes et arbres. Ces données ont servi à évaluer principalement le taux de détection, et la qualité de l'estimation du diamètre à 1m30.

Analyse du taux de détection

Sur l'ensemble des 2875 arbres inventoriés 70.7 % ont été détectés et un diamètre numérique leur a été associé, 17.3 % n'ont pas été détecté mais étaient présent sur le scan et 12.0 % été occultés. Comme expliqué précédemment la quantité d'arbres totalement occultés est en lien direct avec la géométrie de la placette, il n'est donc pas représentatif des performances de Computree. Donc en écartant les arbres totalement occultés et en calculant le nombre d'arbres détectés divisé par le nombre d'arbre potentiellement détectable, le taux de détection est de 80.4 %. Dans la suite de l'analyse le terme d'occultation local ou partiel est utilisé, il signifie que l'arbre est présent sur le scan mais qu'une partie de sa géométrie est occulté par un de la végétation se trouvant entre l'arbre et le lidar.

Concernant l'influence de la structure forestière et l'essence des arbres, la figure n°33 donne la fréquence des types de détection en fonction de la structure forestière, et des essences comptant plus de 40 arbres inventoriés. Cependant il est important de rappeler que les structures forestières, forêt ouverte et futaie irrégulière ne représentent qu'une faible partie des placettes.

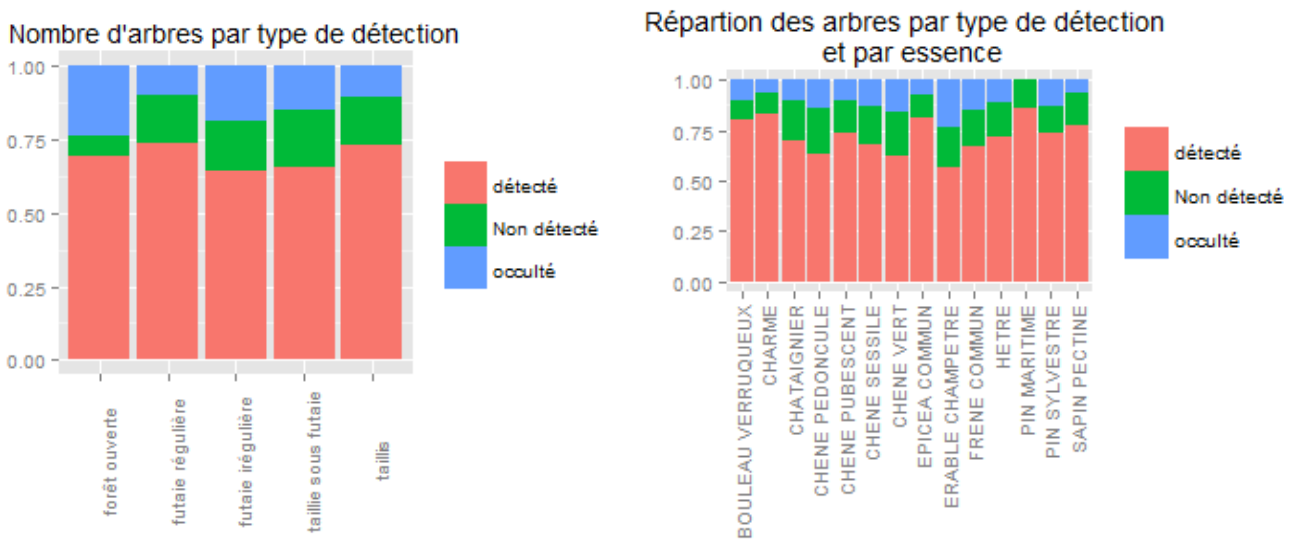


Figure 33 : Fréquence de détection par structure forestière puis Fréquence de détection par essence

Ainsi en futaie régulière le taux de détection est maximum, probablement du fait que l'occultation est minimum (peu de sous étage). La futaie régulière est donc la structure forestière la mieux adaptée au mono-scan, car la disposition régulière et espacée des arbres réduit les phénomènes d'occultation globale. Cependant la proportion d'arbre non détectés est sensiblement identique entre les différents types de structures forestières. Par rapport à l'essence il semblerait que le bouleau verruqueux, le charme, l'épicéa commun et le pin maritime se comportent mieux à l'inverse du chêne pédonculé de l'érable champêtre et du châtaigner. Une des explication du bon taux de détection pour les pins maritime et les épicéas commun est qu'ils sont en grande majorité en futaie régulière issu de plantation. Cependant à ce stade, il est impossible d'apporter une explication au bon ou mauvais comportement de ses essences en observant uniquement ces graphiques.

Concernant les conditions de scan, les deux graphiques de la figure 34 permettent de visualiser la répartition des catégories de diamètres et de la distance ordonnés par type de détection. Selon le protocole d'inventaire les arbres de catégories différentes ne sont par inventoriés dans le même cercle, il est donc nécessaire de dissocier dans l'analyse les arbres par catégorie et par rayon d'inventaire, pour ne pas mélanger l'effet de la distance et du diamètre.

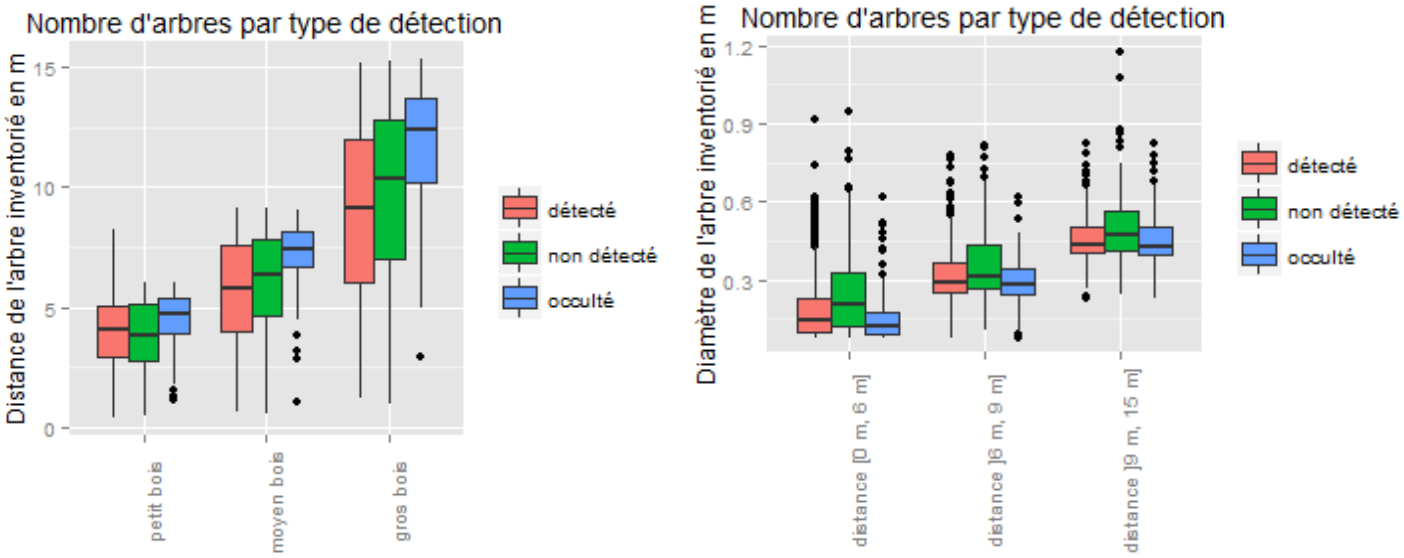


Figure 34 : Graphiques des paramètres de scan

L'observation de la figure permet de confirmer que la distance médiane des arbres occultés est toujours supérieure à celle des arbres non détectés qui est supérieure à celle des arbres détectés. Le choix du rayon de scan est un facteur primordial sur le taux de détection. Concernant le diamètre des arbres inventoriés, il semblerai que les petits arbres soient mieux détectés. Cela peut être expliqué par le fait que les gros arbres sont plus sujets à l'occultation local et donc plus difficilement détectés. Cependant il est important d'analyser plus précisément les caractéristiques des petits arbres avant de porter des conclusions concernant l'influence du diamètre des arbres sur les performances de Computree.

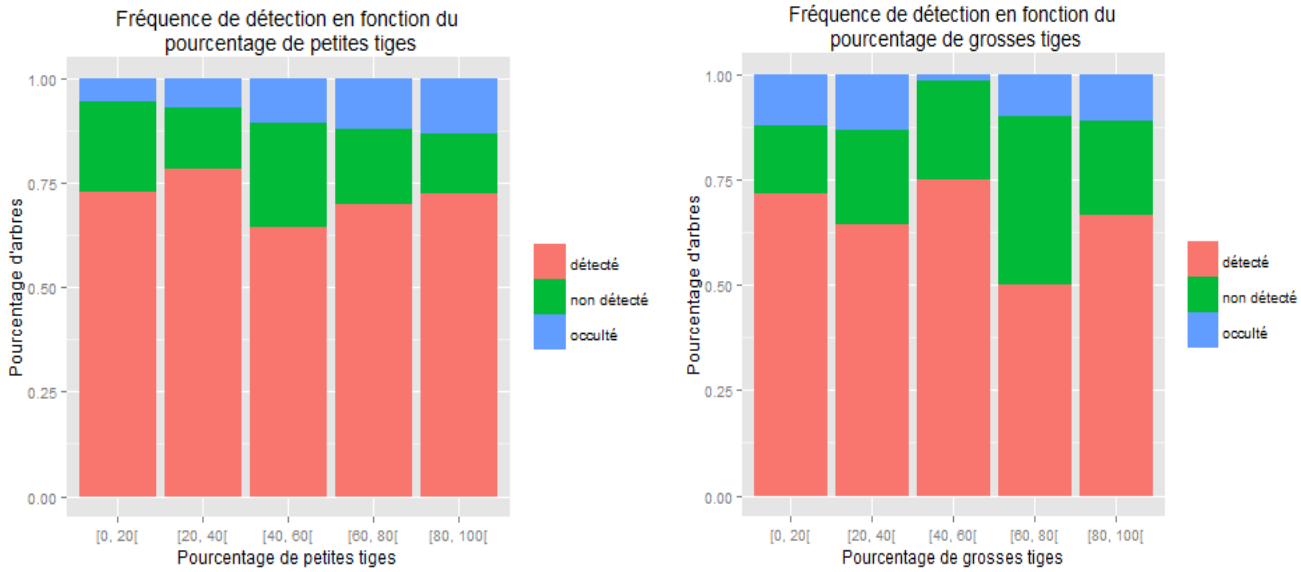


Figure 35 : Graphique de la fréquence de détection en fonction du pourcentage de petites tiges et grosse tiges

En observant le graphique de la figure 35, il est clairement notable que le pourcentage d'arbres occultés augmente avec la proportion de petits bois. Cela s'explique, car les placettes avec de forte densité de petite tiges ont une plus grande probabilité de contenir des cépées ou beaucoup de petits arbres très rapprochés, ce qui génère beaucoup d'occultation total et local et explique les mauvais taux de détection pour ces placettes. D'autant plus que des arbres de faible diamètre ont plus de branches basses ce qui augmente encore l'occultation. Concernant la proportion de gros bois, les arbres de placettes à forte proportion de gros bois ont

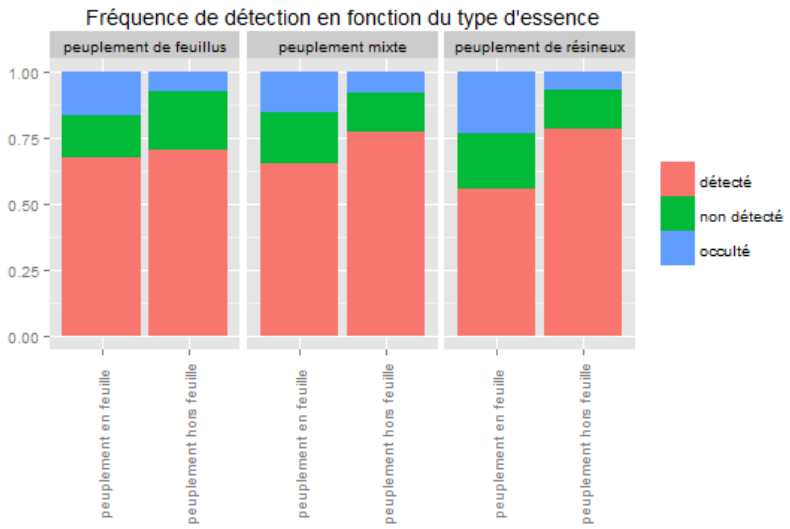


Figure 36 : Graphique de la fréquence de détection en fonction du type d'essence

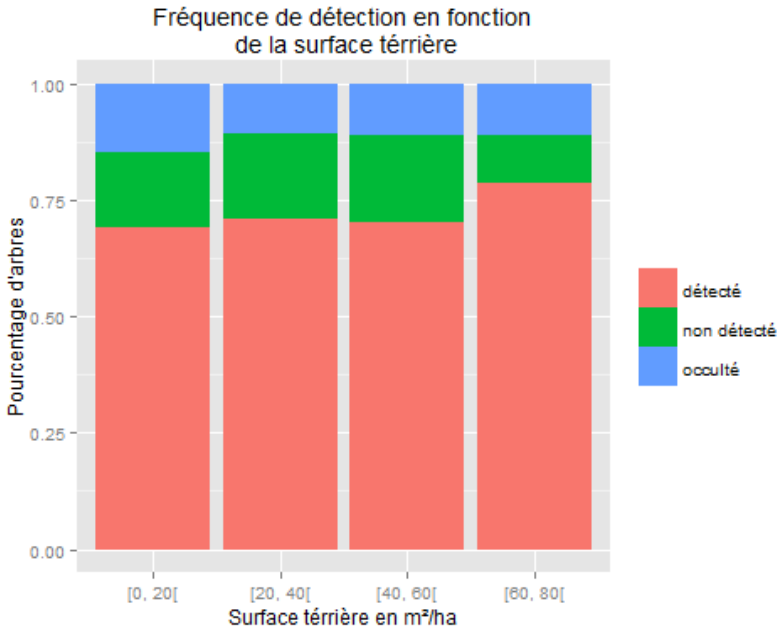


Figure 37 : Graphique de la fréquence de détection en fonction de la surface terrière

beaucoup de non détecter. cela peut s'expliquai par le fait que sur un gros bois il y a une plus forte probabilité que le tronc de l'arbre soit occulté localement par la végétation, hors Computree semble détecter plus difficilement ses arbres avec une occultation partielle.

Concernant les types d'espèces sur des peuplements en feuille est hors feuille, les graphiques suivants (figure 36) présentent les différents taux de détection associés. Premièrement la proportion d'arbres détectés est légèrement meilleure pour les peuplements hors feuille, ce qui semble logique à la vue de l'occultation générée par les feuilles. Dans le cas des peuplements de résineux il y a une forte variation du taux de détection des arbres scannés, selon que la placette est en feuille ou hors feuille. Cela s'explique par le fait que dans les peuplements de résineux, la présence de feuille est liée à la présence d'un sous étage en feuille, or les sous étages génèrent beaucoup d'occultation car ils sont très proches du lidar et donc dégradent énormément le taux de détection. Il est important de noter que la réduction du nombre d'arbres occultés quand on passe de conditions hors feuille à des conditions en feuille se reporte sur la catégorie non détecté, et pas la catégorie détecté. Ainsi les arbres sont visible sur le scan mais il y a une occultation résiduelle qui empêche Computree de bien les détecter. Premièrement cela permet de conclure que la réduction du taux d'occultation est belle est bien efficace en effectuant une mesure en période hors feuille.

Cependant Computree détecte difficilement les arbres avec une occultation locale importante. Deuxièmement la présence d'un sous étage en feuille est un facteur important d'occultation. Troisièmement le type d'essence présent dans le peuplement n'influe que très peu sur le taux de détection.

La figure 37 représente la fréquence de détection en fonction de la surface terrière des placettes. Il est important de rappeler qu'il s'agit de la surface terrière uniquement des arbres inventoriés et qu'elle a été pondérée par le rayon d'inventaire pour chacune des trois catégories de diamètre.

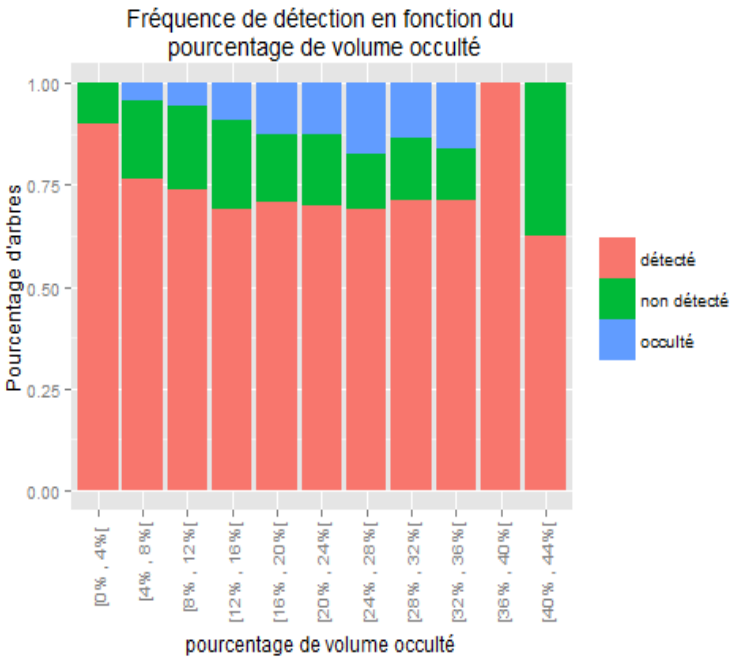


Figure 38 : Graphique de la fréquence de détection en fonction du pourcentage de volume occulté

D'après l'observation de ce graphique, les placettes avec une surface terrière importante ont des quantités d'arbres totalement occultés plus faible. Cela s'explique par le fait que les placettes avec une forte surface terrière sont des placettes généralement avec des arbres de plus gros diamètres plus difficilement occultable complètement par le sous étage. De plus les placette à forte surface terrière ne comporte que rarement un sous étage important, cela explique le fort taux d'arbres bien détectés.

L'indicateur de volume occulté est un indicateur mis en place pour tenir compte du phénomène d'occultation global sur une placette, l'histogramme suivant présente le

taux de détection en fonction de ce dernier

Au vu de la figure 38, la proportion d'arbres occultés augmente au profit des arbres non détectés, alors que la quantité d'arbre détectés reste globalement stable. Cela s'explique par le fait que l'indicateur est représentatif de la densité de végétation, or les placettes avec une fort densité de végétation ont une plus forte probabilité d'avoir des arbres occultés totalement. Cependant les placettes avec un très faible pourcentage de volume occulté ont de très bon taux de détection car une placette avec un très faible pourcentage de volume occulté est une placette très propre, ou les arbres ne sont pas occulté. Dans ces cas, il n'y a que les problèmes liés à la géométrie de l'arbre qui génèrent une quantité d'arbres non détectés. Il y a donc existence d'un seuil en dessous duquel le taux d'occultation total est très faible. Il est difficile d'interpréter les taux de détections pour un pourcentage de volume occulté supérieur 30% car il s'agit d'intervalles contenant un très faible nombre de placette.

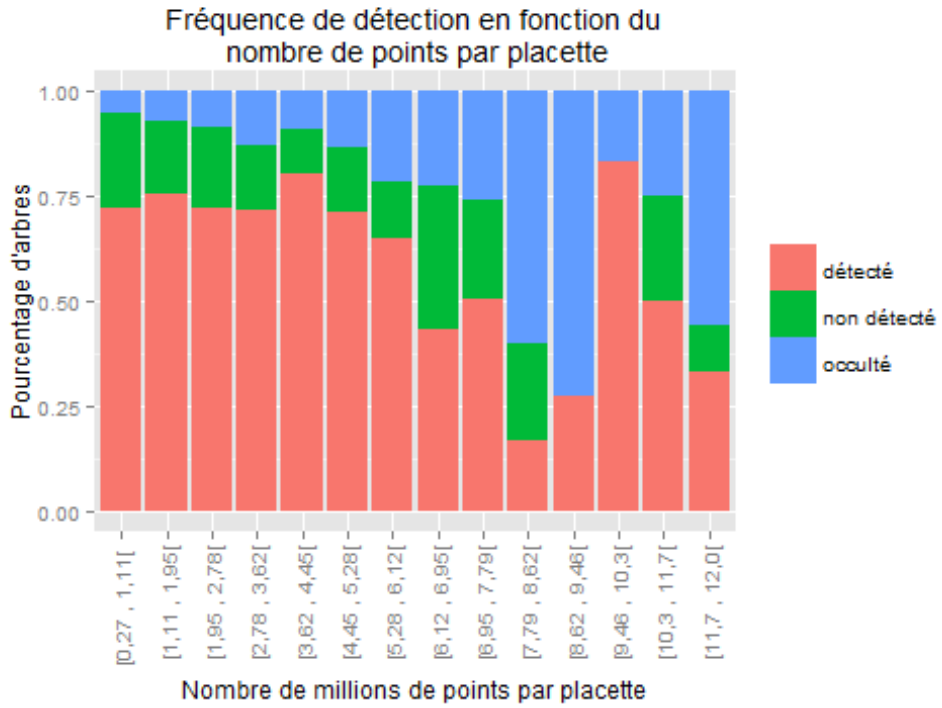


Figure 38 : Fréquence de détection en fonction du nombre de points par placette

La figure 38 donne le taux de détection en fonction du nombre de points par placette. Premièrement il faut préciser que les 4 dernières classes contiennent moins de 10 arbres. Il est donc plus prudent de ne pas interpréter ses valeurs. A la vue de ce graphique il est remarquable que le taux d'arbre totalement occulté augmante d'autant plus que le nombre de points de la placette est important. Ainsi les placettes comportant plus de 5.10^6 points sons des placettes avec une forte occultation et donc un taux de détection des arbres scannés et un taux de détection des arbres inventoriés très faible.

Il est important de ce poser la question de pourquoi le pourcentage de volume occulté dissocie moins les placettes à forte et faible occultation. Une des raisons serait lié au fait que la densité de points de la végétation proche du lidar est très importante, les éléments proches du lidar contiennent donc beaucoup de point. Ainsi l'indicateur de nombre de points renseigne la présence d'une végétation basse dense, or ce type de végétation génère un occlusion important sur l'étage dominant comportant les arbres inventorié. L'indicateur de pourcentage de volume occulté lui est moins sensible à la présence d'une végétation basse car cette végétation n'occulte pas beaucoup de volume à comparait du volume qu'un arbre pourrait occulté. Ainsi l'indicateur de volume occulté mélange l'effet lié à la présence d'une végétation basse et l'effet lié a une forte densité d'arbres de l'étage dominant. Or le taux d'occlusion semble beaucoup plus dépendant de la présence d'un sous étage, c'est pourquoi l'indicateur de nombre de point est plus pertinent pour représenter l'évolution des taux de détections des placettes.

En conclusion le taux de détection ne sembla pas être influencé par la structure forestière, à l'exception d'un taux d'arbres occultés légèrement supérieur pour les taillis sous futaie. Le type d'essence et les condition feuille hors feuille ne semble pas non plus beaucoup influencé le taux de détection. En revanche dans le cas de placette avec une forte surface terrière le taux d'occultation est bas, et il semblerait que le nombre de points par placette explique relativement bien l'évolution du taux de détection. La présence d'un seuil pour l'indicateur de pourcentage de volume occulté a été noté, en dessous duquel il n'a quasiment pas d'arbres occultés, cependant il serait intéressant d'étudié plus finement l'existence de ce seuil dans le cadre d'une prochain évaluation de Computree.

Analyse du diamètre estimé à 1m30

A présent il convient d'évaluer la qualité d'estimation du diamètre sur la partie des arbres inventoriés pour lesquels un diamètre a été estimé. Dans la suite de l'analyse le terme erreur de diamètre est la différence entre le diamètre à 1 m 30 estimé par computree et le diamètre mesuré en forêt, en gardant à l'esprit que l'erreur de mesure au ruban est d'environ +- 1cm sur le diamètre. La figure 40 représente les diamètres estimés en fonction des diamètres inventoriés, la droite identité est tracée en bleu. Il est observable qu'une faible minorité des arbres ont un diamètre largement sous estimé. Mais cependant une grand majorité des point ce trouve proche de la droite identité.

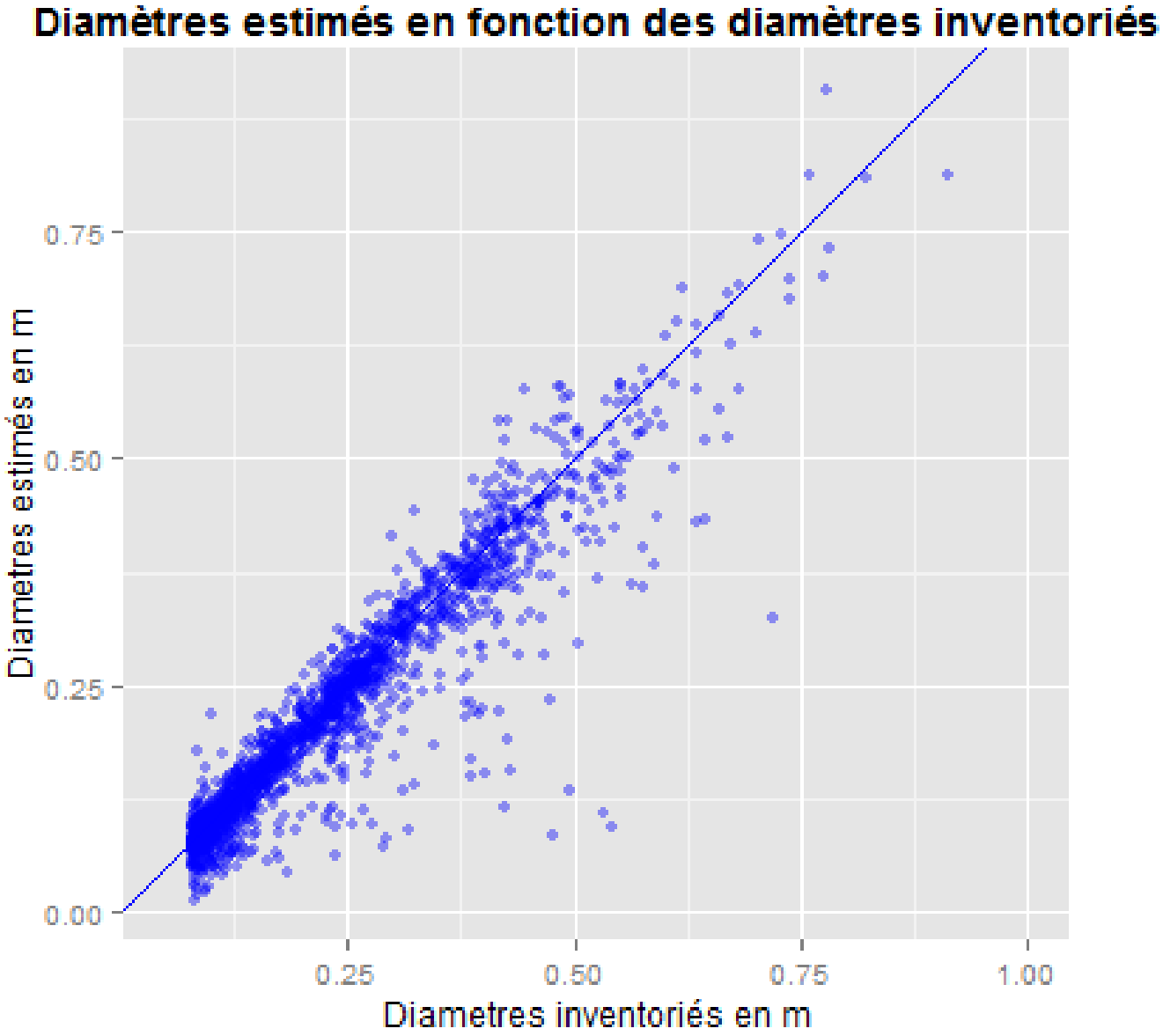


Figure 36 : Graphique des diamètres estimés en fonction des diamètres inventoriés

La figure 40 donne l'histogramme de distribution des erreurs. La médiane en rouge est légèrement inférieur à zéro, la distribution est donc légèrement décalé . Les deux droites noires sur la figure 41 représentent le 1er décile et le 9^{ième} décile. Il est observable qu'une minorité de la distribution sous estime beaucoup le diamètre inventorié et s'étale sur la « gauche ».

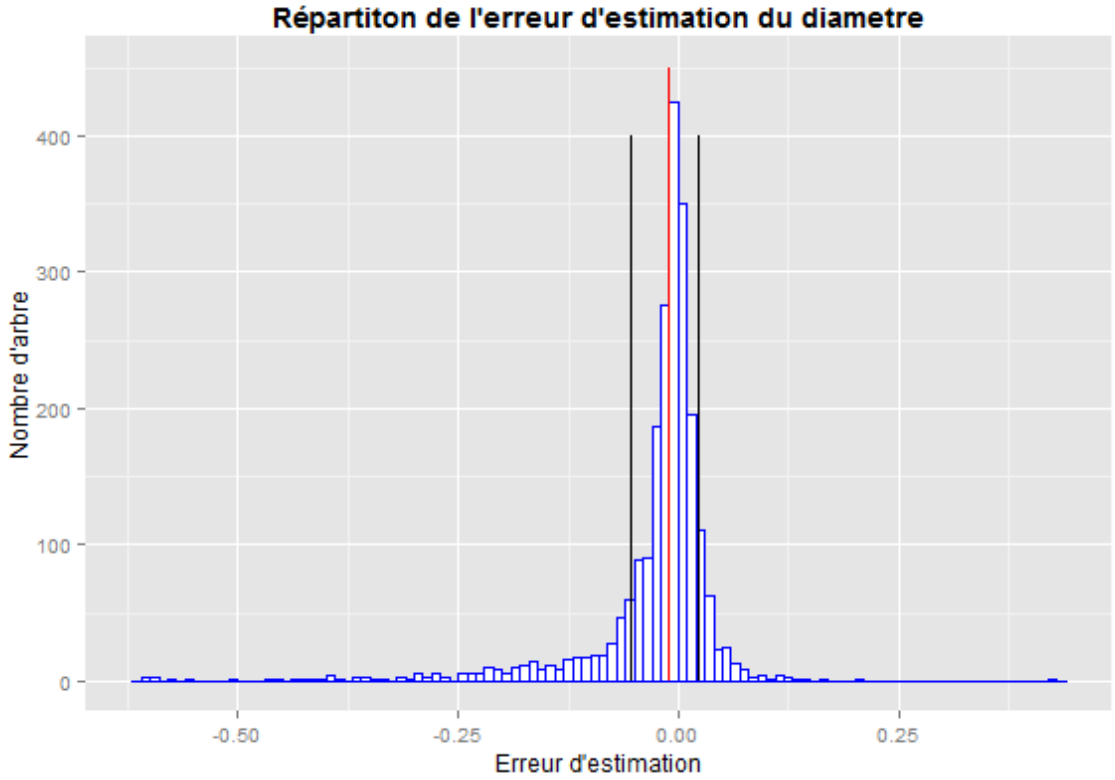


Figure 40 : Histogramme de la distribution des erreurs du diamètre

Il est ainsi clair qu'une minorité des points et fortement sous-estimée, il est donc important de filtrer ces outliers avant d'analyser plus en détail l'influence des différents critères sur la population d'arbre « normalement » traité. Le seuil choisie pour écarter les outiliers et donnée par la table de Student en écartant un point pour mille. Le seuil est de 3,293 fois l'écart type. Le tableau de la figure 41 donne les écarts des statistiques sur l'erreur de diamètre des deux populations d'arbres.

	Population non filtré	Population filtré
Ecart type	4,48 cm	3,21 cm
Moyenne	-1,27 cm	-0,85 cm
Médianne	-0,55 cm	-0,50 cm

Figure 41 : Tableau résumé des statistiques de l'erreur de diamètre

En analysant la population d'arbre filtré, il est remarquable sur la figure 42 qu'il y a un biais dans l'estimation du diamètre. Les diamètre sont ainsi d'autant plus sous-estimé que l'arbre est loin du lidar.

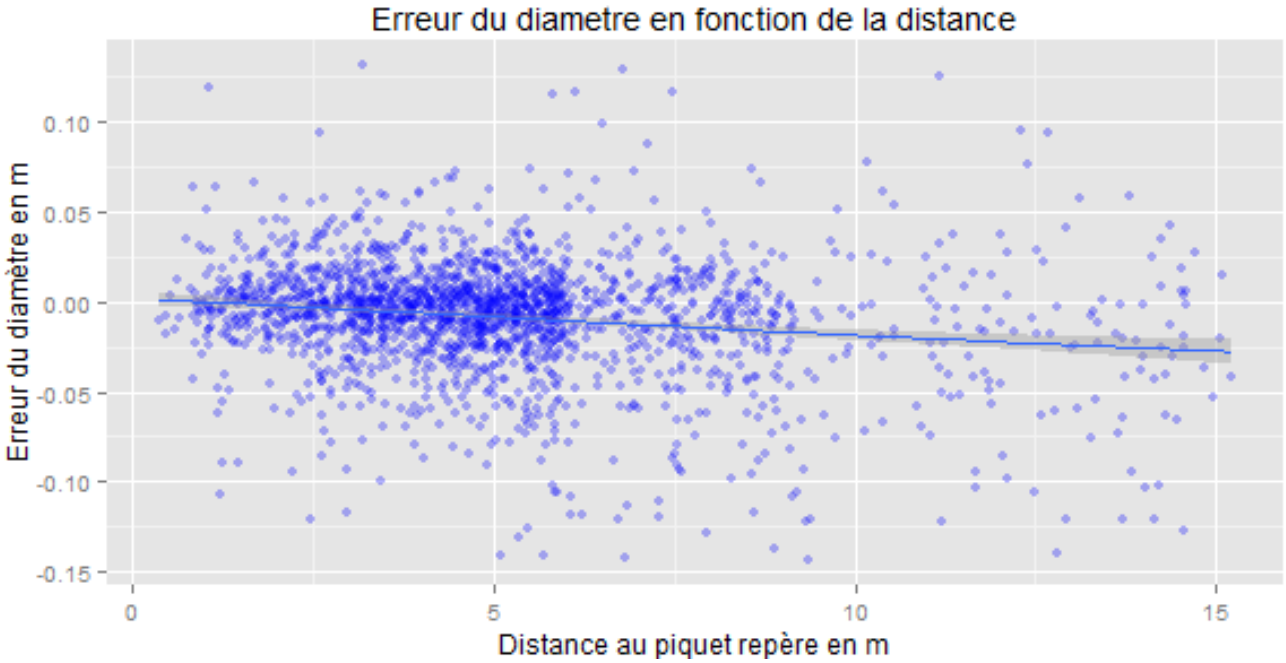


Figure 42 : Graphique de l'erreur d'estimation du diamètre en fonction de la distance

Il est difficile de porter une explication claire à ce biais. Il se pourrait que ce biais provienne de la manière d'ajuster les cylindres aux Clusters. En effet les cylindres sont ajustés par critère des moindres carré, le cylindre passe ainsi au milieu des points. Une méthode plus proche de la mesure de circonférence de terrain serait de prendre le plus petit cylindre contenant tout les points.

Le graphique de la figure 43 présente l'évolution de l'erreur de diamètre en fonction du diamètre de l'arbre. Il confirme bien la même tendance à sous estimer le diamètre.

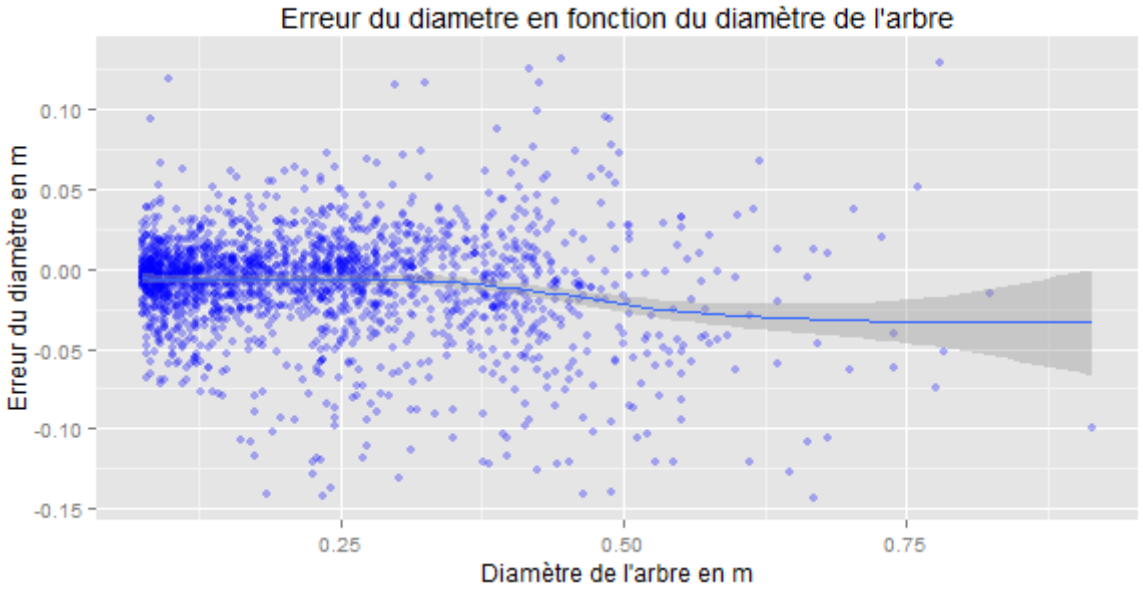


Figure 43 : Graphique de l'erreur de diamètre en fonction du diamètre des arbres

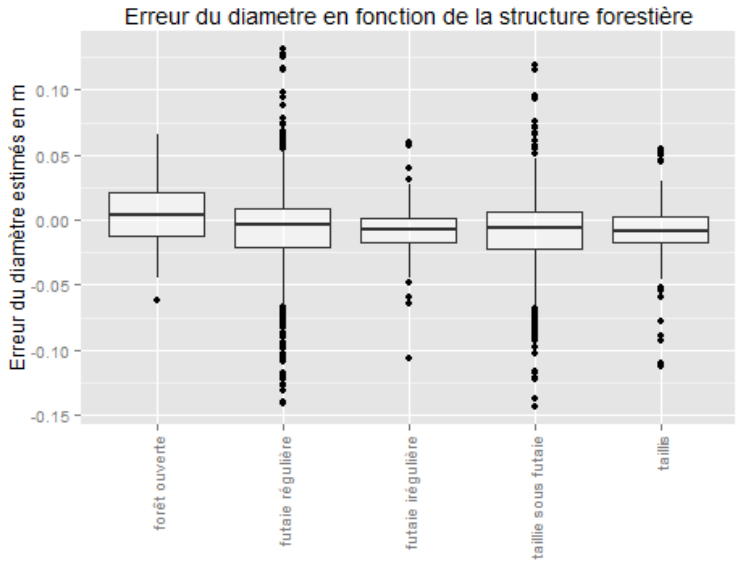


Figure 44 : Graphique de l'erreur de diamètre en fonction de la structure forestière

Il convient maintenant d'analyser l'influence de la structure forestière, du type d'essence et la présence de feuille sur l'estimation du diamètre. En visualisant les différentes boites à moustaches de la figure 44 , il semblerai qu'il n'y est pas de différence majeur entre une futaie régulière et un taillis sous futaie. Les types forêts ouverte, futaie irrégulière et taillis ne disposant d'un nombre d'arbres réduit, il est prudent de ne pas conclure quand à ces types de structure forestière. Après avoir fait la vérification de l'égalité des variances et de distribution normale sur les deux sous populations (futaie et taillis sous futaie) une analyse de la variance a été mené pour tester si la structure forestière génère un biais sur l'erreur d'estimation du diamètre. Le tableau de la figure 45 confirme la non existence d'un biais.

	ddl	Somme des carrés	Carré moyen	F	P.value
Structure forestière	1	0.0001	0.000082	0.078	0.78
Résidu	1791	1.8885	0.001054		

Figure 45 : Tableau récapitulatif de l'ANOVA de structure forestière entre futaie et taillis sous futaie

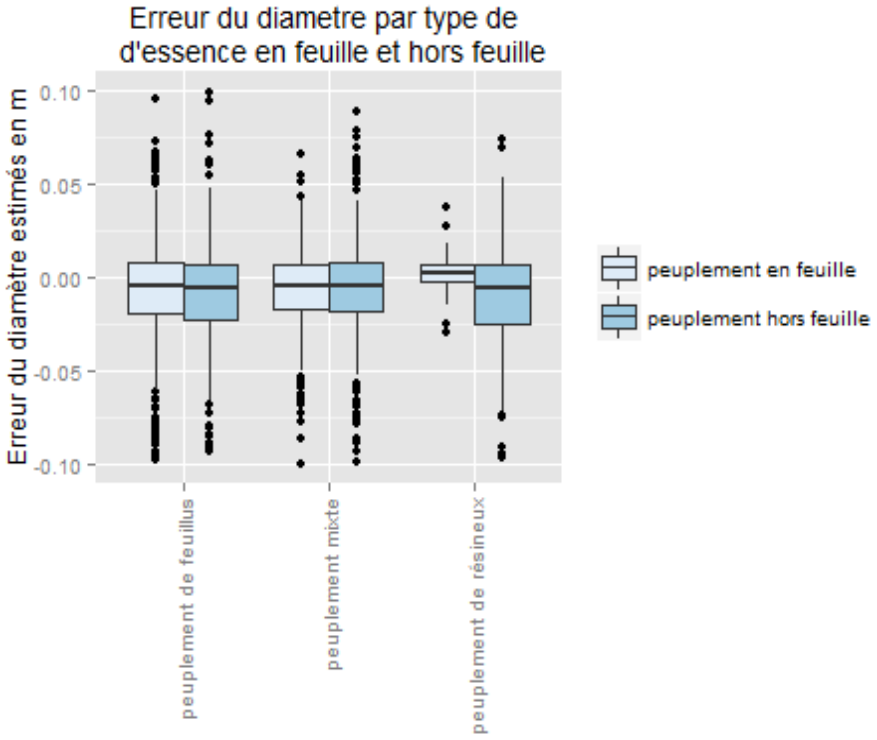


Figure 46 : Graphie de l'erreur d'estimation du diamètre en fonction du type de peuplement

Par rapport à la distinction des peuplements contenant uniquement des feuillus ou uniquement des résineux ou un mixte de résineux feuillus, la figure 46 permet d'observer la répartition de l'erreur d'estimation du diamètre. Il semblerai que le type d'essence n'est pas une facteur influent sur l'estimation du diamètre. Il est notable que pour les peuplements de résineux, l'erreur médiane est plus faible pour les peuplements en feuille que hors feuille. Cela peux s'expliquer par le fait que le nombre de placettes de résineux en feuille, c'est-à-dire un peuplement de résineux avec un sous étage de feuillu en feuille est très faible. Il serait plus pertinent à l'occasion d'une prochaine évaluation de Computree de distinguer

la présence ou non d'un sous étage en feuille ou non. Finalement le diamètre est légèrement mieux estimé pour des peuplement de feuillu et mixte en feuille que hors feuille. L'une des raison est que dans ses peuplements en feuille les arbres ont une plus forte probabilité d'être totalement occultés et ses arbres ne sont pas pris en compte dans l'estimation du diamètre. Alors que dans les peuplement de feuillu et mixte hors feuille le taux d'arbres partiellement occultés et plus fort, hors c'est pour des arbres partiellement occulté que Computree génère les plus grosses erreurs de diamètre. Une ANNOVA sur le type d'essence et et la présence de feuille est résumé à la figure 47.

	ddl	Somme des carré	Carré moyen	F	P-value
Feuille/Hors feuille	1	0.0005	0.0004775	0.452	0.501
Résineux/Feuillu	2	0.0002	0.0000886	0.084	0.919
Résidu	1789	1.8879	0.0010553		

Figure 47 : Tableau récapitulatif de l'ANOVA type de peuplement

L'analyse de la variance confirme bien que l'estimation du diamètre n'est pas biaisé par le type d'essence des arbres, ou par la présence de feuille.

Une étude similaire aux critères structure forestière et type d'essence a été mené sur les essences comptant plus de 50 arbres. Il semblerai que l'appartenance à une essence en particulier n'est pas un critère influant sur la capacité de Computree à bien estimer le diamètre à 1m30.

Des indicateurs ont été mise en place afin de pouvoir détecter les situation où les algorithmes de Computree ne sont pas performants. Il est possible de classer les indicateurs en place en trois classes. La première classe est représentative du manque d'information géométrique, la deuxième de la structure numérique de l'arbre détecté par computree et la dernière met en évidence les problèmes de formes. Afin de mieux distinguer la pertinence de chacun des indicateurs, pour la suite de l'analyse, il s'agit de la population d'arbres non filtrés qui est utilisée.

Indicateur représentatif d'un manque d'information géométrique

Les graphiques de la figure 48 et 49 représente, pour le premier l'erreur du diamètre en fonction du nombre de points de la section (ensemble des clusters associé à un même arbre) et pour le deuxième l'erreur du diamètre en fonction de l'indicateur de densité de point.

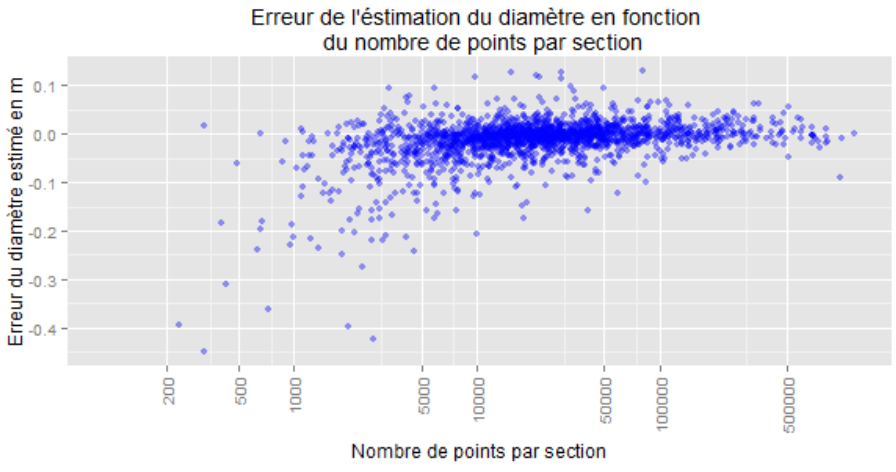


Figure 48 : Graphique de l'erreur du diamètre en fonction du nombre de point par section

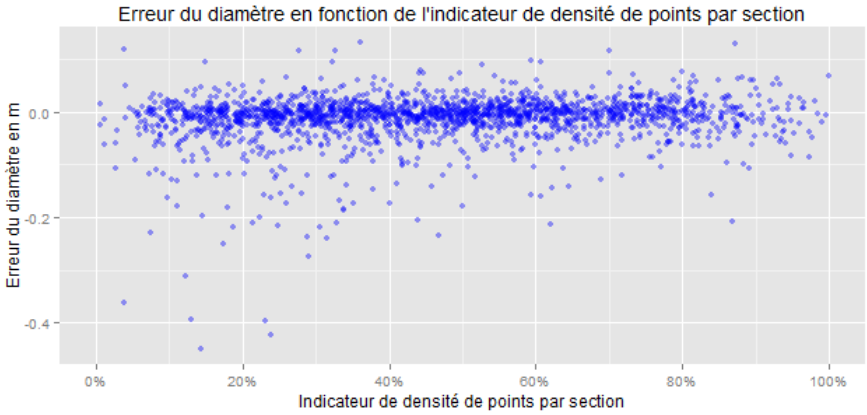


Figure 49 : Graphique de l'erreur du diamètre en fonction de l'indicateur de densité de points

L'indicateur le plus discriminant est donc le nombre de points par section, il permet de localiser deux seuil. Le premier ,à environ 100.000 points, au dessus duquel la précision est très bonne, et un deuxième de 5000 point en dessous duquel la probabilité de sous-estimé le diamètre à 1 m 30 est plus forte. L'indicateur de densité de point représente le nombre de point de la section divisé par le nombre de point maximal théorique. Le fait que cet indicateur n'est pas aussi explicatif, démontre que l'erreur de diamètre n'est pas uniquement lié à la densité de point, mais aussi à la quantité de point. En effet l'indicateur de densité de point, n'est pas faible pour une petite section contenant peu de points, alors que l'ajustement des cylindres sur une petite section conduit généralement à une erreur plus importante du diamètre. Contrairement à l'indicateur de nombre de points par section, qui vas être faible pour une grande section avec une forte occultation local mais aussi pour des petites sections.

Indicateur représentatif de la structure numérique de l'arbre

Pour rappeler brièvement la structure numérique des arbre dans Computree, chaque arbre est décrit par une section. La section est composée de différents clusters d'une hauteur de 10 cm. Les clusters sont constitué de points, et leurs est associé un barycentre et un cylindre dans les cas ou les cylindres sont correctement ajusté. Ainsi les indicateurs choisis pour expliquer les performances de Computree sont le nombre de cylindre par section et le nombre de clusters par section. Les Figures 45 et 46 présente l'évolution de l'erreur de diamètre en fonction du nombre de clusters puis du nombre de cylindres.

Comme expliqué dans les parties précédentes les arbres numériques sont constitués de clusters et de cylindres. Pour une tranche horizontal donnée , un arbre a un nombre maximum de cluster et de Cylindre. Les figures suivantes donnent l'impact du nombre de cluster et cylindre sur la qualité d'estimation du diamètre.

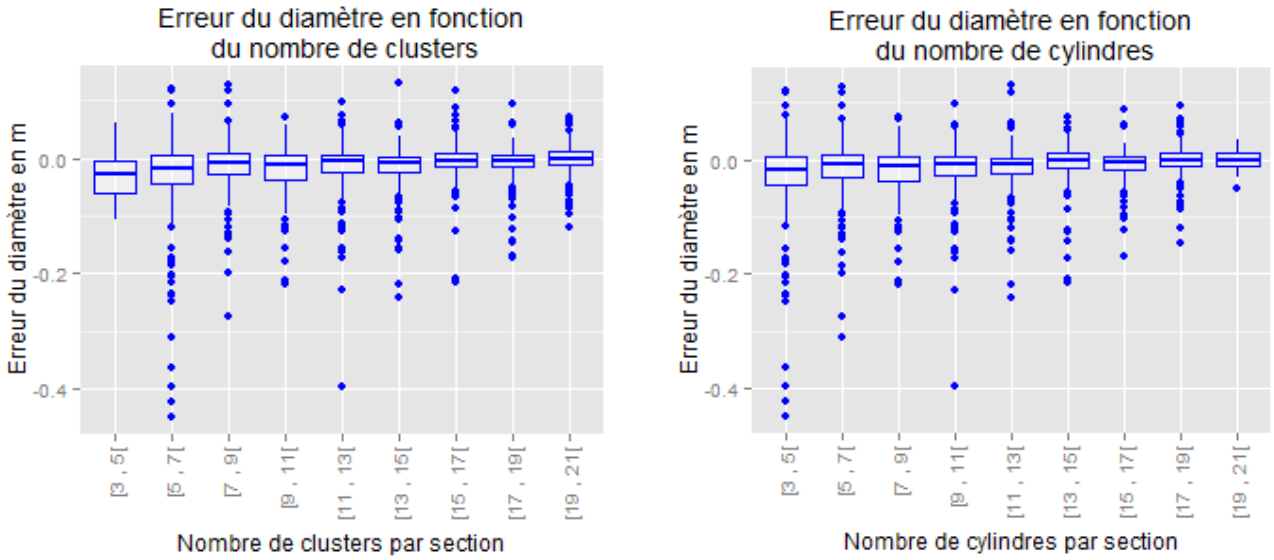


Figure 37 : Graphique de l'erreur d'estimation du diamètre en fonction du nombre de clusters puis en fonction du nombre de cylindres

En visualisant la figure 50, la phénomène de sous-estimation des diamètres pour des arbres comportant peu de cylindres ou de cluster est confirmé encore. Les deux indicateurs sont représentatif d'une dispersion plus marquée de l'erreur de diamètre pour un seuil de moins de 7 éléments. Il est important de souligner que dans cette évaluation les arbres ont été pris en compte entre 30 cm et 2 m 30, ainsi comme les clusters sont de 10 cm le maximum d'éléments est de 20 par arbres (21 dans quelques cas). C'est-à-dire que les arbres disposant de 20 éléments sont décris géométriquement sur toute la hauteur. Cependant les cylindres sont filtrés par critère de taille, inclinaison et de qualité d'ajustement, donc pour deux arbres avec le même nombre de clusters celui qui a le plus de cylindres est le mieux détecté. Cela explique le fait que pour l'indicateur du nombre cylindres la dispersion de l'erreur se ressert au niveau du maximum d'élément. Donc l'indicateur de cylindre discrimine mieux l'erreur de diamètre à 1m 30.

Indicateurs représentatif de la forme de l'arbre

Les indicateur de formes du squelette des barycentre des clusters et des cylindres est représentatif de la forme géométrique de l'arbre. La figure 51 donnent l'erreur du diamètre en fonction de l'indicateur de forme du squelette des barycentre puis en fonction de l'indicateur de forme des centre des cylindres ajustés.

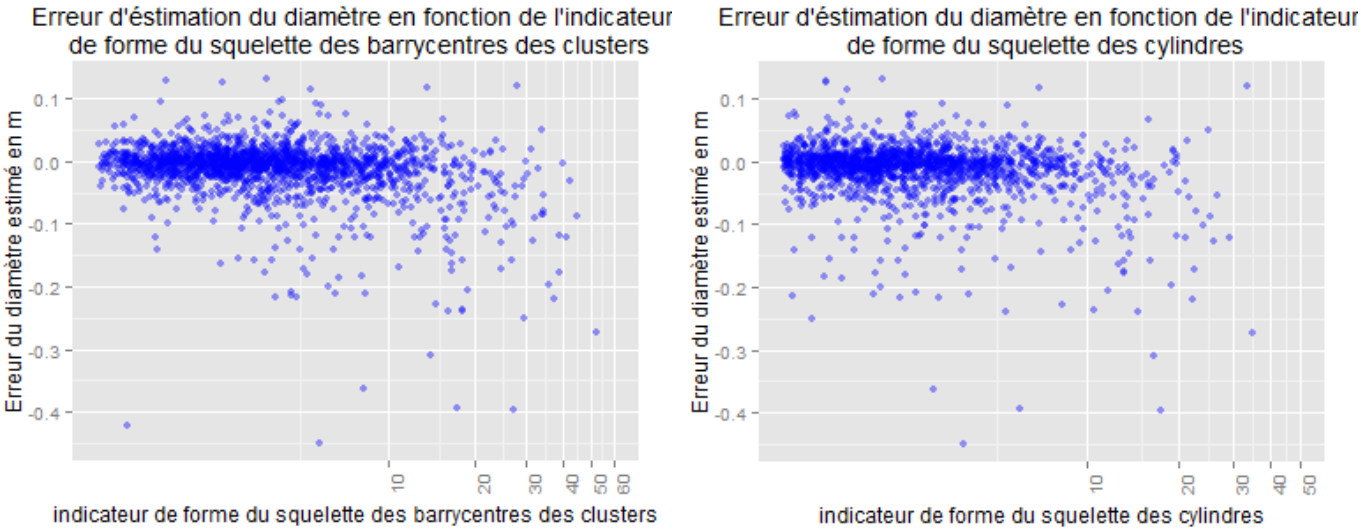


Figure 51 : Graphique de l'erreur d'estimation du diamètre en fonction de l'indicateur de forme des barycentre des cluster puis en fonction des squelette des cylindres

Pour rappel, l'indicateur de forme et la somme de la valeur absolue des angles du squelette moins l'angle total de l'arbre divisé par le nombre d'élément, ainsi un arbre avec une courbure très chaotique aura un indicateur élevé. L'indicateur de forme du squelette des cylindres semble très peu discriminant, en revanche l'indicateur de forme du squelette des clusters dissocie mieux les bonnes et mauvaises évaluations du diamètre à 1 m 30. Un seuil bas au environs de 2° délimite un zone ou Computree est très précis, et un seuil haut de 10°, pour lequel la probabilité de mal estimé le diamètre est un peut plus forte.

Le tableau de la figure 52 suivant récapitule les indicateurs à utiliser pour chaque classe d'indicateur avec les seuil associé.

Classe d'indicateur	Nom de l'indicateur	Seuil de très bonne estimation du diamètre	Pertinence du seuil bonne estimation du diamètre	Seuil de mauvaise estimation du diamètre	Pertinence du seuil mauvaise estimation du diamètre
Information géométrique	Nombre de point par section	> 100 000 points	***	<500 points	**
Structure Numérique	Nombre de cylindre par section	19 cylindres	***	<7 cylindres	**
Forme de l'arbre	Indicateur de forme du squelette des barycentres	<30°	**	>50°	*

Figure 52 : Tableau récapitulatif des seuil des indicateurs

Pour conclure sur la qualité d'estimation du diamètre, premièrement une grande majorité des arbres détecté ont un diamètre estimé aavec un écart au diamètre mesuré de l'ordre de 4cm. Il y a une légère tendance à sous estimé légèrment le diamètre qui s'explique par la technique d'ajsutement des cylindres mais qui peut se corriger facilement en modifiant les algorithmes, il y a une minorité d'arbre qui est fortement sous estimé liée à des problèmes d'occultation ou de formes de l'arbre. Deuxièment les critères de structure forestière, essence,

condition de feuille hors feuille ne sont pas influant sur la qualité d'estimation du diamètre, à l'inverse de la distance ou un biais est clairement identifiable. Finalement l'analyse des différents types d'indicateurs à fixé les indicateurs les plus explicatifs, ainsi que des seuil de bonne et mauvaise détection.

b)Analyse qualitative des résultats

Grace à l'analyse statistique un certain nombre d'arbres avec une estimation du diamètre aberrante ont été détectés. La visualisation de ces arbres et de leurs structure a permis de classifier les problèmes récurrents, ils sont présentés dans le schéma de la figure 53.

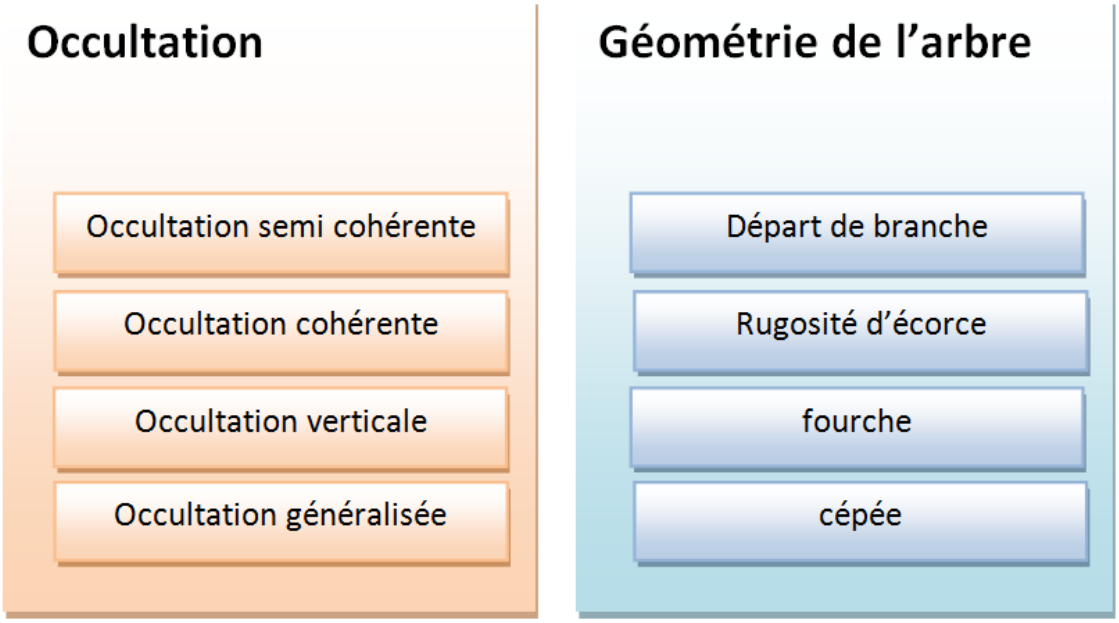


Figure 53 : Schéma de classification des problèmes

Problème d'occultation

L'occultation est donc le premier grand type de problème rencontré. Il est subdivisé en 4 catégories. L'occultation dite semi cohérente est de l'occultation générée par des branches de petits diamètres ou des feuilles. Elle affecte la section en supprimant un petit nombre de points de manière cohérente. L'occultation dite cohérente est générée par un arbre sur un autre arbre, ou le relief dans de cas rares. Elle affecte la section en supprimant un grand nombre de points de manière cohérente. L'occultation vertical est un cas particulier de l'occultation cohérant, elle peut supprimer un grand nombre de points et scinder un arbre en deux sections verticales. L'occultation généralisée supprime un nombre de points moyen, sans cohérence particulière, souvent rencontré sur des arbres loin du lidar.

Le problème principal de la segmentation semi cohérente est qu'elle segmente l'arbre en une multitude de sections différentes comme présenté sur la figure 54. La figure montre que pour une occultation cohérente de faible distance la section n'est pas segmentée. En fait d1 est inférieur au paramètre distance d'un point à un groupe de l'étape Horizontal Clustering. Alors que d2 et d3 sont supérieur a cette distance, donc pour un même arbre à une hauteur donnée deux clusters lui est associés, ceci vas rend plus difficile la fusion des sections entre elles. Finalement plusieurs sections sont

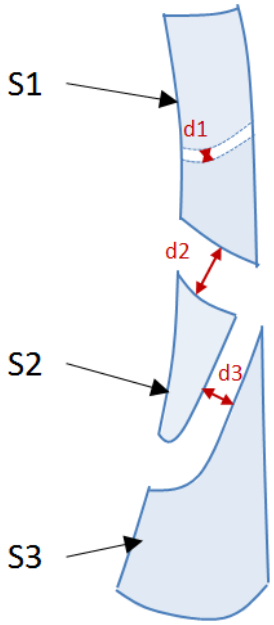


Figure 54 : Schéma représentatif de l'occultation semi cohérente

associé aux même arbre et il faut donc choisir manuellement qu'elle section représente le mieux l'arbre. Les captures d'écran illustrent l'influence de l'occultation semis cohérente sur la structure géométrique d'un arbre.

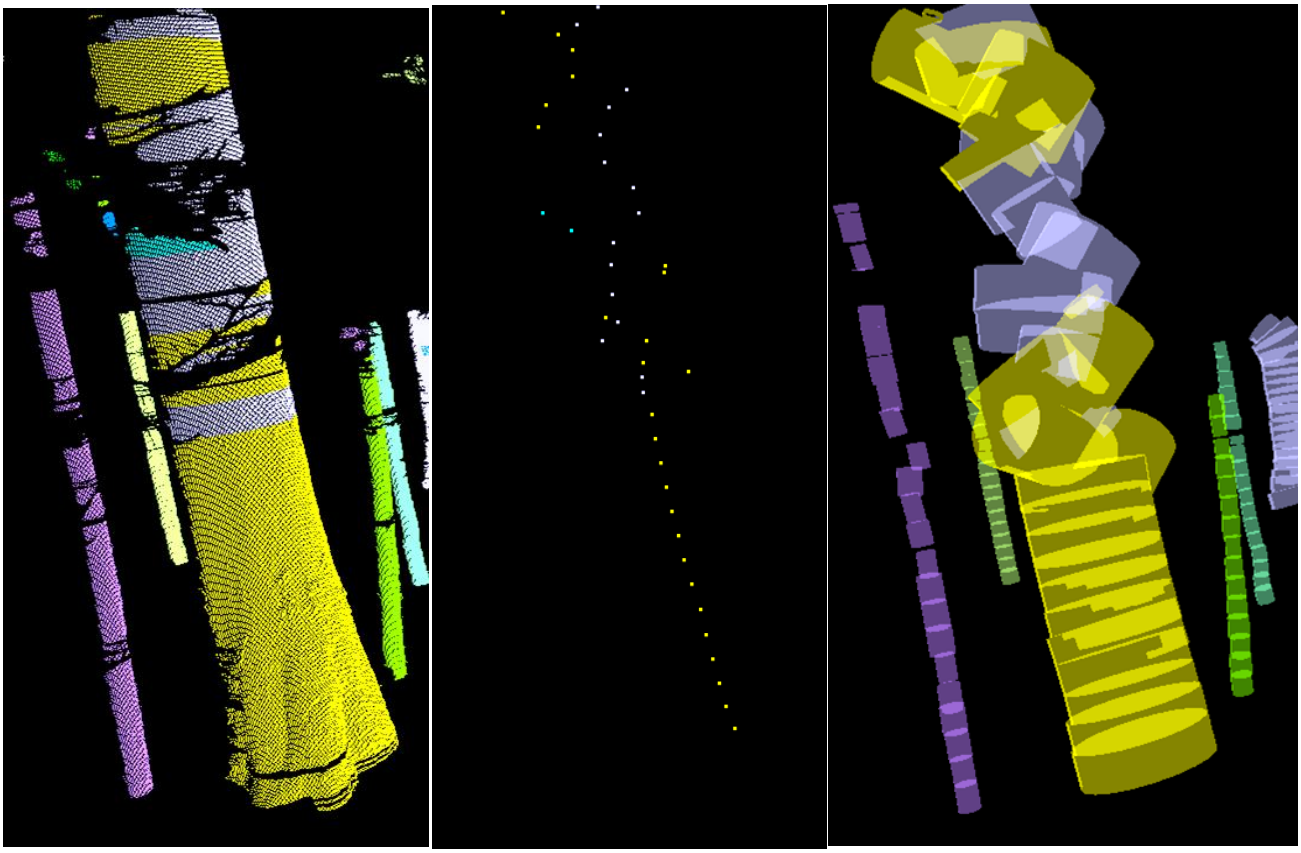


Figure 55 : Capture d'écran de l'occultation semis cohérente (à gauche les points des section, au milieu les barycentres des clusters, à droite les cylindre)

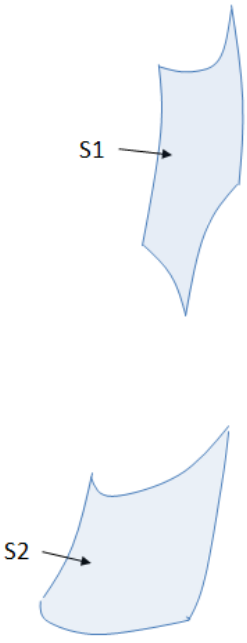


Figure 56 : Schéma de l'occultation cohérente

Sur l'exemple de la figure 55, l'occultation semi cohérente a conduit à une segmentation de l'arbre, qui a engendré des squelettes de barycentre « tordu » et comme l'axe principale des cylindres est formé par le squelette des barycentres, les cylindres générés sont mal ajustés. Hors l'étape qui détermine le diamètre à 1 m 30 supprime automatiquement les sections dont la décroissance métriques des cylindres est supérieur à un certain seuil, ainsi le diamètre de l'arbre est soit mal estimé du fait de la mauvaise estimation des cylindres, soit l'arbre n'est pas détecté.

L'occultation cohérente a l'inconvénient majeur de supprimer une partie importante de l'information géométrique, de plus quand elle est situé au milieu de l'arbre (comme sur la figure 56), la fusion des sections est alors plus délicate. Cependant il n'y a pas de segmentation des sections, du fait de la continuité de l'occultation, par contre le squelette de barycentre est fortement décalé par rapport à l'axe central de l'arbre, cela génère un ajustement des cylindres faussé, comme on le remarque sur les captures d'écran figure 57. Une sous-estimation marqué est produite pour les cylindres ajustés sur les clusters proche de l'occultation. Computree interprète cette différence de taille de cylindres (étape : extract diametre from cylinder) comme un arbre aberrant et le supprime, il y a

donc dans ce cas des problèmes de détection ,ou, dans le cas où l'arbre n'est pas écarté, une sous-estimation du diamètre à 1m 30.

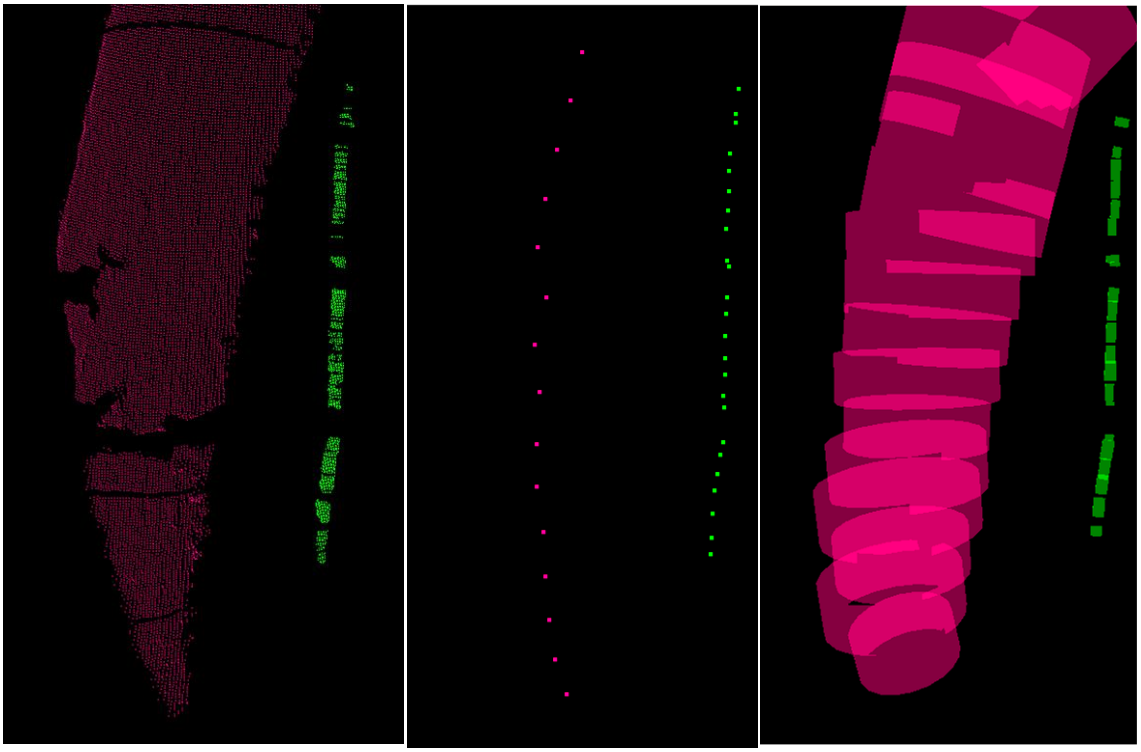


Figure 57 : Capture d'écran de l'occultation cohérente (à gauche les points des section, au milieu les barycentres des clusters, à droite les cylindre)

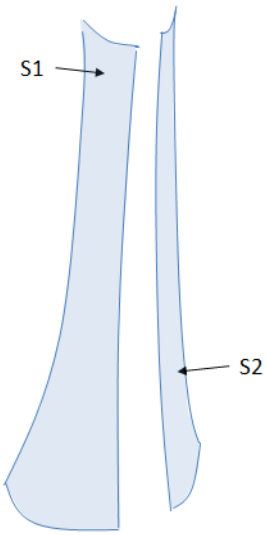


Figure 58 : Schéma de l'occultation vertical

L'occultation verticale (voir figure 58) est gênante quand l'étape de création des clusters détecte deux clusters au lieu d'un pour chaque tranche horizontale d'un même arbre, et que l'étape de fusion des sections voisines ne fonctionne pas. Malgré le fait que le paramètre de distance de recherche des sections voisines est paramétrable, augmenter la distance de recherche augmente le risque de fusionner les petits arbres proche les uns des autres, ou de fusionner des branches verticales avec la tige principale.

La capture d'écran (voir figure 59) illustre le phénomène d'occultation vertical vue de dessus. Dans ce cas la distance entre les deux sections était trop grand . Donc pour le même arbre, Computree vas reconstruire deux arbres numériques. Encore une fois une intervention manuelle est nécessaire pour choisir l'arbre généré depuis la section la plus grande. Le deuxième inconvénient est que l'occultation verticale réduit significativement la portion d'arc sur laquelle sont ajustés les cylindres. La portion d'arc étant plus plane la précision d'ajustement des cylindres est réduite.

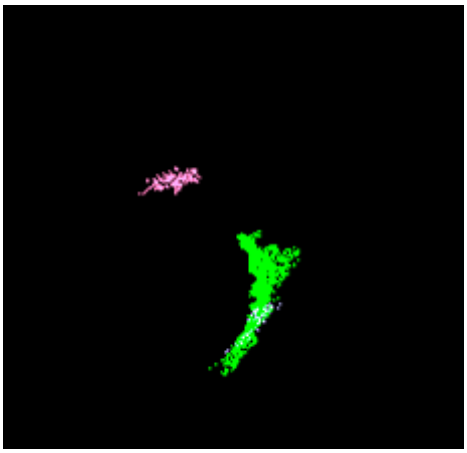
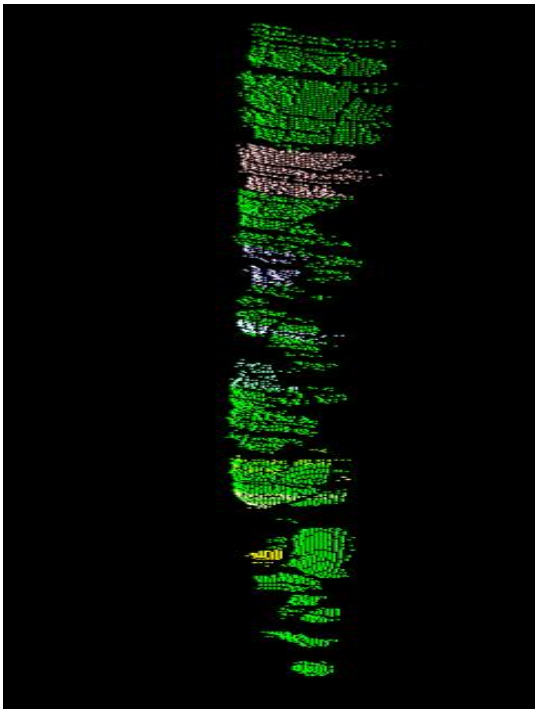
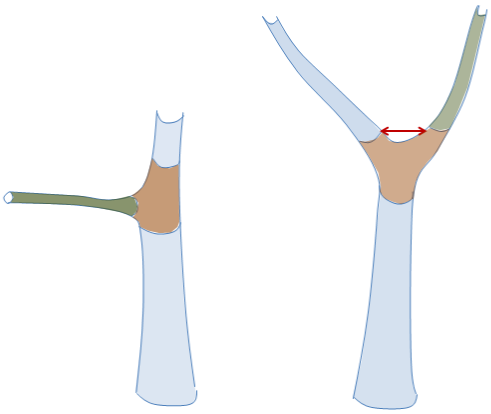


Figure 59 : Capture d'écran d'occultation verticale vue de haut



La perte d'information géométrique liée à l'occultation généralisée (exemple figure 60) n'est pas systématiquement pénalisante pour la détection de l'arbre ou l'ajustement des cylindres. Dans le cas d'une occultation généralisée homogène sur la section les conséquence sont pratiquement nul. Mais dans les cas ou l'occultation perturbe beaucoup le squelette des barycentres, les cylindres seront mal ajustés et ainsi l'étape d'extraction du diamètre à 1m 30 risque de filtrer l'arbre ou de lui associer un diamètre aberrant. Il y a aussi un risque de segmentation de l'arbre. La captures d'écran illustre ce phénomène.

Figure 60 : Capture d'écran d'occultation généralisé



Problème de géométrie de l'arbre

Un départ de branche et une fourche sont deux types de problèmes similaires, le premier problème est que Computree doit détecter la partie de l'arbre qui est la tige principale, le deuxième problème est lié à la perturbation de la circonférence de l'arbre dans ces zones. En effet la zone orange sur la figure 61 n'est généralement pas cylindrique et comporte une forte variation du diamètre.

Figure 61 : schéma de départ de branche et de présence de fourche

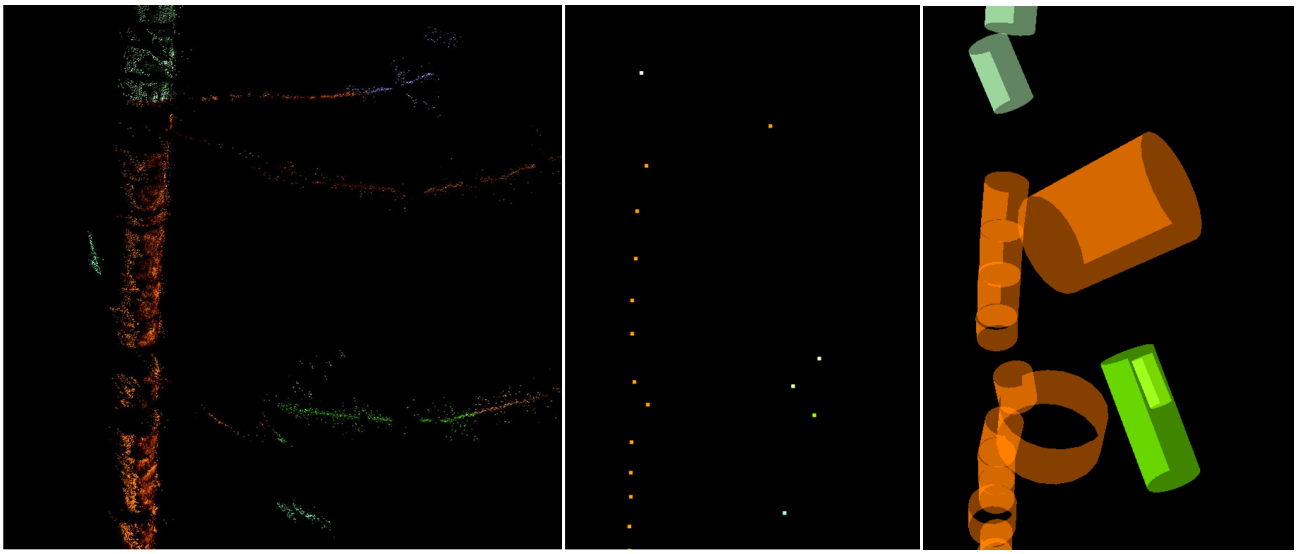


Figure 38 : Captures d'écran d'un départ de branche (à gauche les points, au milieu les barycentres des clusters, à droite les cylindres)

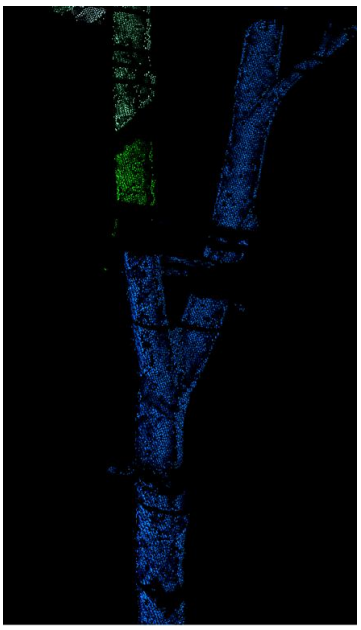


Figure 63 : Capture d'écran d'une fourche

Les captures d'écran de la figure 62, illustrent les deux cas d'un départ de branche. Quand la branche est correctement identifiée, comme la branche verte l'arbre conserve une unique section, la zone de transition est associée à l'arbre et décale le squelette de barycentre et génère des cylindres surestimés. Dans le cas ou la branche est associé à l'arbre, le squelette de barycentre est fortement perturbé, ainsi le haut de l'arbre n'est pas associé au même arbre numérique. Les deux cas handicapent l'intégrité de la structure de l'arbre et de l'estimation du diamètre. Les algorithmes de construction et de fusion des sections devront mieux tenir compte du squelette de barycentre des clusters.

La capture d'écran de la figure 63 illustre le cas d'une fourche. Dans ce cas ci la zone de transition est complètement englobée par l'arbre bleu et même après la fourche les deux tiges sont dans la même section. Cela est très pénalisant pour l'ajustement des cylindres et conduit à des estimations de diamètre aberrantes.

Pour certaine espèce la rugosité de l'écorce est très marquée, cela est une contrainte pour l'étape de filtrage des polylignes . Pour résumer le fonctionnement de cette étape, sur chaque Cluster horizontal de 1cm de haut, des particules sont réparties de manière homogène sur les points du cluster, ensuite une polyligne relie les particules entre elle, cette polyligne est alors filtré si elle ne correspond pas à un arc de cercle, le cluster est alors filtré. La figure 64 présente la répartition de ses différents éléments géométriques.

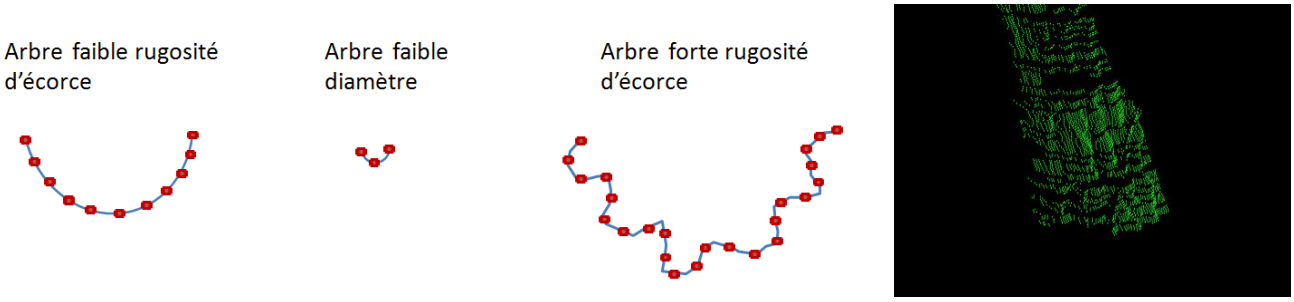


Figure 64 : Schéma explicatif du problème de rugosité et capture d'écran associée

Sur la figure 64 en bleu est tracé le cluster horizontal, en rouge les particules, par soucis de lisibilité la polyligne n'a pas été tracé. La distance entre deux particules est paramétrable. La première contrainte est de disposer de au moins 4 points sur les petits arbres, afin de détecter si il s'agit d'une portion d'arc de cercle ou d'une forme géométrique aléatoire, cela impose une distance maximum entre les polylignes. Le filtrage prend en compte la distance moyenne entre un cercle ajusté sur les particules et les particules, si cette distance moyenne dépasse un seuil le cluster est filtré. Ainsi les arbres à faible rugosité et les petits arbres ne sont pas filtrés. Les arbre à forte rugosité eux ont en moyenne des particules plus écartées du cercle ajusté, et donc il sont filtrés. La solution pour pallier à ce problème est d'augmenter la distance entre les particules afin de lisser le profile des particules, en d'autre termes de gommer la rugosité. Cependant en augmentant la distance des particules on rend le filtre inefficace sur les petits clusters. Il s'agit donc de faire le choix de conserver le filtrage sur les petits clusters et de perdre les clusters des arbres avec un écorce rugueuse ou l'inverse. La capture d'écran illustre que certain cluster horizontaux on été quand même filtrés, cependant la majorité de la section a été conservé.

Les cépées ont l'inconvénient majeur d'avoir une forte densité de petites tiges verticales très rapproché les une des autres. Il s'agit du problème inverse de l'occultation verticale, il faut que Computree détecte que deux sections proches l'une de l'autre sont deux arbres différent. Ici encore il s'agit d'un compromis entre le choix de distinguer deux petits arbres proches l'un de l'autre et le choix de fusionner deux sections découpées verticalement. Pour distinguer deux arbres d'une cépée, il faut que la distance de recherche d'un point à un cluster et que la distance de fusion des sections voisine soient inférieur à la distance entre les deux arbres. Il s'agit bien d'un choix car en diminuant ses distances on augmente la segmentation des arbres avec des problèmes d'occultation. Ce problème est illustré avec l'arbre violet de la capture d'écran de la figure 65. Effectivement les pieds des différents arbres de la cépée appartiennent tous au même arbre, puis deux tiges sont affecté au même arbre. Il est donc dans ce cas impossible à Computree d'estimer un diamètre correct.

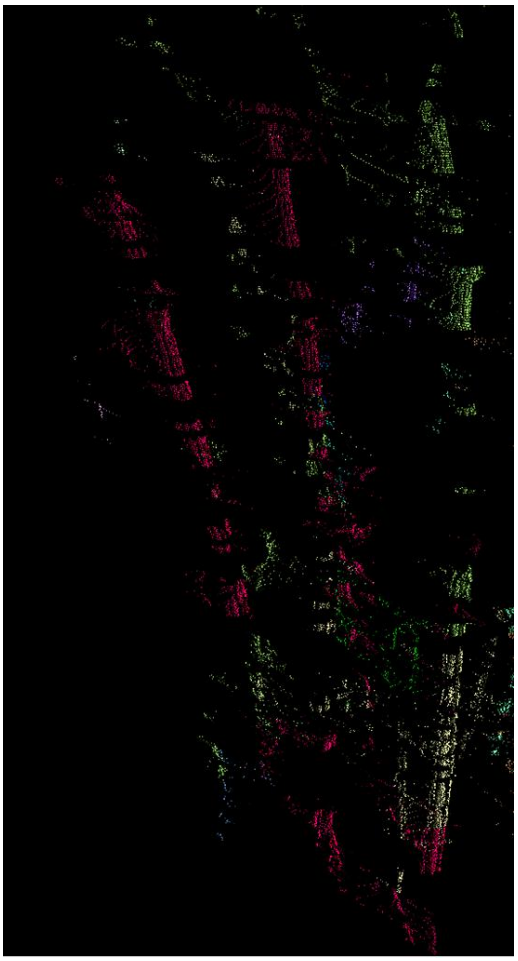


Figure 65 : Capture d'écran d'une cépée

Un problème qui peux aussi se rencontrer sur une cépée est quand deux tige se croise et que Computree se trompe dans l'identification des arbres comme sur la capture d'écran de la figure 66.

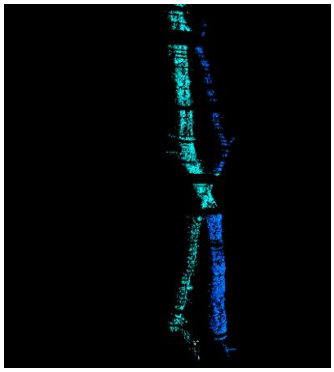


Figure 66 : Capture d'écran d'une cépée croisée

A la suite de cette analyse qualitative il a été mis en place un sous échantillons de scans représentatif de chacun des ses problèmes, afin d'évaluer les modifications d'un algorithmes plus rapidement et plus efficacement. Le tableau de la figure 67 présente les placettes choisies et leur problème associé.

Nom placette	Caractéristique
Lyon 2011 66510604	Plusieurs cépées présentes
Montpellier 2011 67410774	Départs de branche
Nancy 2011 67410774	Présence de fourches
Bordeaux 2011 28110736	Futaie régulière plus appareil aérien visible
Bordeaux 2011 43120826	Futaie régulière
Lyon 2011 67110616	Occultation cohérente et semi cohérente
Lyon 2011 67110616	Occultation généralisé
Lyon 2011 67110616	Occultation verticale

Figure 67 : Tableau de présentation du sous échantillon

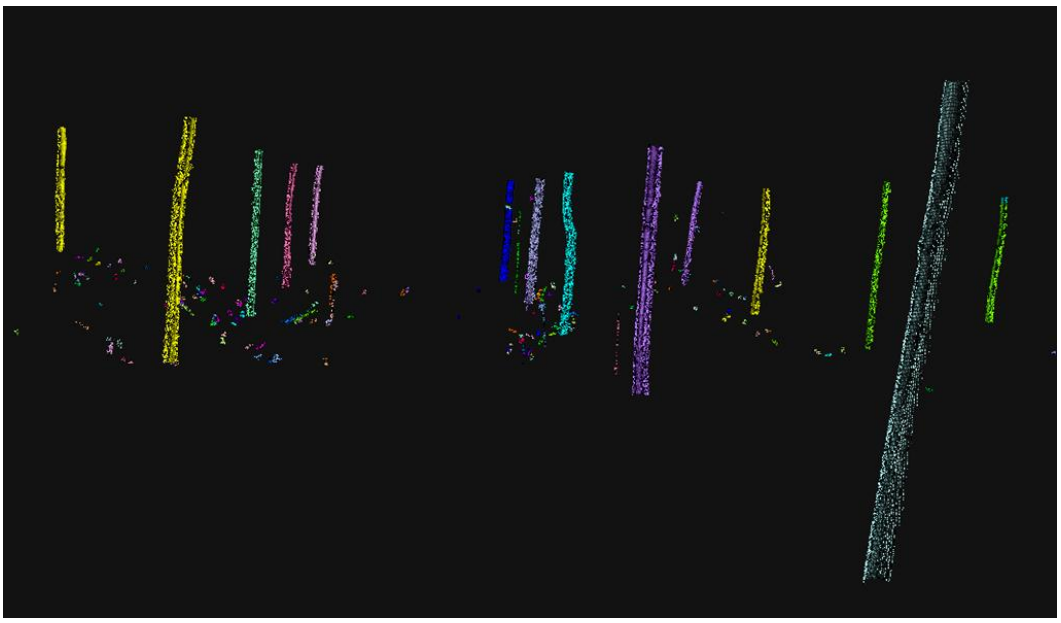


Figure 68 : Capture d'écran d'une futaie régulière coupé entre 30 cm et 2m 30 très bien détecté!!!

4)Conclusion

A la suite de l'analyse statistique et qualitative il est possible de conclure sur les points suivants ;

Pour une majorité des arbres faiblement occultés, avec une géométrie simple sur un intervalle de 30 cm à 2 m 30, Computree détecte convenablement l'arbre et lui associe un diamètre à 1m 30 avec une erreur inférieur à 4cm, comme illustré sur la figure 68. La proportion d'arbres totalement occultés augmente avec la distance. La proportion d'arbres occultés est aussi sensible à la structure forestière. Il est donc envisageable dans de futur protocoles d'augmenter le nombre de scans pour des peuplements en taillis sous futaie. Malheureusement le temps disponible au cour de cette évaluation fut trop court pour évaluer l'influence du multi scan. Il serait intéressant toutefois d'évaluer l'influence du nombre de scans sur le taux d'arbres occultés au cours d'une prochaine évaluation de Computree. Au cours de l'analyse statistique Il a été démontré que la présence de sous étage ou de végétation impacté très sensiblement le taux de détection des arbres scannés et le taux de détection des arbres inventoriés, il serait pertinent de mettre en place une échelle pour quantifier la densité du sous étage afin d'étudier sont influence sur les performances de Computree.

Concernant les indicateurs explicatifs, le nombre de points par placette permet de prédire des taux de détection très bas quand il dépasse un certain seuil. Le nombre de cylindre et le nombre de points par arbre quand lui en dessous d'un certain seuil, indique qu'il y a une plus grande probabilité que Computree sous-estime les diamètre des arbres. De même quand l'indicateur de forme du squelette des cylindres excède un certain seuil, Computree a plus tendance à mal évaluer le diamètre à 1m30. Il a été noté aussi que l'estimation du diamètre n'est pas influencé par la distance de l'arbre, le type de structure forestière et le type d'essence. De manière général, encore aucun indicateur n'a été trouvé pour expliquer complètement l'erreur sur l'estimation du diamètre, en revanche il est possible de définir des seuils pour lesquelles la probabilité que Computree sous-estime les diamètres est plus grande.

Lors de l'analyse qualitative un certain nombre de problèmes liés à l'occultation et à la géométrie des arbres a mis en évidence l'importance du paramétrage des étapes de traitement. Généralement un type de paramétrage réduis les problèmes d'occultation et augmente les problèmes liés à la détection d'arbres et inversement. De plus cette analyse montre que les performances de Computree sont très sensibles à la forme du squelette, aux distances de recherche de la création de clusters initiaux, de fusion des sections voisines et de la distance de répartition des particules. La difficulté de paramétrage et d'élaboration d'algorithmes s'expliquent en partie car

Computree est développé dans une optique de gestion. Ainsi la diversité des peuplements et la complexité de la géométrie des arbres engendrent un grand nombre de cas particuliers qui doivent être résolus sans pour autant dégrader les performances de Computree dans des cas simples. Ainsi une première classification des problèmes et la mise en place d'un sous échantillon donnent les outils nécessaires à l'élaboration d'étapes de traitement efficaces et stables vis-à-vis de la complexité des cas rencontrés.

En résumé Computree est déjà performant pour une grande majorité des cas, il reste toutefois un fort potentiel d'amélioration pour traiter convenablement les cas atypiques. Cependant les choix de conception de la plateforme Computree permet d'intégrer très facilement de nouveaux algorithmes et de les tester sur un grand nombre de scans.

Figure 39 : Figure de remerciement aux agents de l'IGN

Bibliographie

Vos bois mode d'emploi Michel Hubert Institut pour le développement forestier 1997 (livre)

COLIN Aurélie Evaluation de Computree, logiciel d'inventaire forestier à partir de scans laser terrestres : Mémoire de stage, Master Fage, Nancy, 2011,32p

MAILLET Cédric Conception et constitution d'une base de données de scans laser terrestres et de données forestières de validation : Rapport de stage, Licence Professionnelle S.I.G,2012,p94

TRICOT Etienne Reconstitution d'inventaires forestiers avec le scanr laser terrestre LiDAR : Rapport de projet de fin d'étude, Diplôme ingénieur Arts et Métiers PARITECH,2010,p85

SUMNALL M.J, Hill R.A et Hinsley, S.A. SilviLaser 2012, The estimation of forest inventory parameters from small-footprint waveform and return discrete airborn LiDAR data In : Septembre 2012, Vancouver, Canada

MORNEAU François, HERVE Jean-Christophe. Note technique : Tarifs de cubage à l'Inventaire forestier national, Inventaire Forestier National, Juin 2010, p14

La forêt en chiffres et en cartes IGN. Service de l'inventaire forestier national éd 2012. France, 32p

Rapport d'activité 2010 Inventaire Forestier National, Service de l'inventaire forestier national éd 2010. France, 72p

Lexique

Aménagement (forestier) : l'aménagement d'une forêt est un document qui, sur la base d'une analyse préalable de la forêt, de son environnement économique et social et de sa gestion passée, fixe les objectifs à atteindre et planifie, pour une durée de 15 à 25 ans, les interventions en coupes et en travaux dans le but de garantir la gestion durable d'une forêt bénéficiant du régime forestier.

Donnée dendrométrique : termes de sylviculture associé à des mesures géométriques des arbres, comme par exemple la hauteur, le diamètre à 1 m 30.

Espace des tangentes : Il s'agit d'un espace en deux dimensions où chaque vecteur est représenté par un point, l'abscisse du point est la norme du vecteur et l'ordonnée est l'angle du vecteur par rapport à un vecteur de référence. Dans le cas de notre étude les vecteurs en question sont des portions de polylignes représentant des portion d'arcs de cercles.

Futaie irrégulière : peuplement comportant des arbres d'âges différents et où se juxtaposent dans le temps et à l'échelle de l'unité de gestion, des opérations de régénération et d'amélioration.

Futaie régulière : peuplement comportant des arbres sensiblement du même âge - et du même diamètre - à l'échelle de l'unité de gestion (parcelle, sous-parcelle), ce peuplement étant issu de régénération naturelle ou de plantation (exceptionnellement de rejets : futaie sur souche).

Multi-scans : Afin de limiter les phénomène d'occultation il est possible d'effectuer plusieurs scans de la même placette depuis des endroits différents. Cependant il est nécessaire de disposer des sphère de calage afin de faire la fusion des scans entre eux.

MNT : Modèle Numérique de Terrain, ou représentation de la topographie d'une zone terrestre.

Peuplement (forestier) : ensemble d'arbres croissant sur une surface déterminée.

Structure (d'un peuplement) : elle est appréciée au niveau de l'unité de gestion, en fonction de l'éventail des classes de diamètres significativement représentées sur l'unité. On distinguera deux types de structures : la structure régulière et la structure irrégulière.

Surface terrière d'un arbre (ou d'un peuplement) : superficie de la section de la tige (ou des tiges) mesurée à 1,30 m du sol. La surface terrière, ramenée à l'hectare et exprimée en m² a pour symbole "G". C'est un paramètre très important en foresterie, il renseigne sur l'importance du couvert, la concurrence entre les arbres et le capital sur pied. Très facile à mesurer sur le terrain

Taillis : peuplement formé de tiges issues de rejets de souches (par opposition à la futaie composée d'arbres en général issus de semis de franc pied coupées régulièrement à faible diametre).

Taillis-sous-futaie : peuplement forestier constitué d'un taillis simple surmonté d'une futaie d'arbres d'âges variés.

Tarif : ou tarif de cubage. Il s'agit d'un tableau fournissant le volume des bois sur pied ou abattus à partir de leur mensuration (en général diamètre et hauteur).

Tables des Figures

Annexe I

Pré-traitement des données de lidar terrestre utilisé pour l'évaluation des performances de Computree

Ce document a été rédigé afin de présenter les étapes de pré-traitement effectuées durant le stage, qui pour certaines représentent plusieurs jours de travail. L'objectif est de faire le point sur l'ensemble des opérations de pré-traitement nécessaires et de pointer les difficultés rencontrées. Un certain nombre de recommandations sont évoquées dans la dernière partie, en espérant que ce document contribuera à l'amélioration de la qualité du jeu de placette numérisé par l'IGN et facilitera ainsi sont utilisation.

I. Etapes de pré-traitement des données lidar

1) Dépouillement des fiches terrains

Lors du dépouillement des fiches terrains, les informations essentielles relevées pour l'utilisation des données étaient les suivantes :

- Le nom du scan central
- L'orientation de départ du scan central
- Les positions des scans périphériques
- Les orientations de départ associées aux scans périphériques
- L'identifiant de la placette

Les informations secondaires potentiellement pertinentes relevées étaient :

- La pente de la placette
- Les conditions climatiques (intensité du vent, présence de neige, de pluie, de grêle, de brouillard)
- La date du scan
- Noter si le peuplement est en feuille ou hors feuille
- Noter si la végétation basse de la placette a été dégagée
- Les remarques des agents à propos d'une situation exceptionnelle

2)Conversion des fichiers faro en fichier xyb

Lors de la conversion des fichiers, les fichiers enregistrés par le lidar (format .fls) ont été ouverts avec le logiciel Faro Scène©, une rotation de l'orientation de départ du scan a été réalisée, puis les fichiers ont été exportés en format .xyb. Le format .xyb est un format binaire libre facilement utilisable par d'autres programmes, notamment Computree.

3) Ajustement de l'orientation des scans par rapport à l'inventaire

Lors de cette phase chacun des fichiers .xyb a été ouvert sur Computree. Parallèlement l'inventaire terrain a été importé sur Computree permettant de visualiser à la fois les nuages des points et les cercles représentant la position des arbres à 30cm du sol. Ainsi il fut facile de recaler les nuages de points exactement sur les inventaires. L'objectif de cette phase de pré-traitement est de faciliter l'étape d'association automatique entre les arbres numériques détectés par Computree et les arbres inventoriés.

II. Difficultés rencontrées lors de la phase de pré-traitement

L'ensemble des difficultés rencontrées lors de la phase de pré-traitement et d'utilisation des données sont listées ci-dessous :

- L'absence de fiche terrains pour certains scans, ou la non correspondance entre les identifiants placettes sur les fiches terrains et les noms des fichiers de scans.
- Un mauvais positionnement de départ du lidar lors de la mesure, lié au fait de la symétrie de l'appareil, il arrive que les agents le pose à l'envers, ainsi un décalage de 180° est observé lors de la visualisation du scan. Cela implique lors de la première rotation de détecter si il y a un décalage entre les données inventoriés et le fichier .fls.
- L'impossibilité de détecter si le scan est décalé de 180° ou non, du fait d'une végétation basse masquant les arbres inventoriés.
- Une erreur de mesure de l'orientation de départ, impose cette fois ci de recaler le nuage de point sur Computree pour corriger cette erreur. Il s'agit généralement d'une erreur de rotation de +/- 5 GRADs.
- Lors de l'utilisation des données, la non correspondance entre les identifiants des noms de scan avec les données d'inventaires type structure forestière, essence des arbres, circonférence à 1m30.
- Détérioration de la qualité des scans par la présence d'un agent, de la mallette du lidar ou d'outils de mesures sur le scan.

III. Recommandation pour la phase d'acquisition et de pré-traitement des données

La première recommandation qui semble logique au vu de la liste des difficultés rencontrées est l'utilisation d'un identifiant unique, pour les fiches terrains, les noms de scans et les données d'inventaires. De plus l'ensemble des recommandations ci-dessous pourrait améliorer la qualité du jeu de placettes scannées :

- Etablir une convention pour nommer les placettes comportant un code de localisation du centre rattaché à la placette, l'année de scan et l'identifiant placette. Il est intéressant d'utiliser plusieurs champs pour caractériser le fichier placette. Cela permet de le retrouver plus rapidement quand il est rangé dans un dossier contenant beaucoup de scans. A noter qu'il est plus performant lors d'une utilisation automatique des données que chacun des champs du nom garde le même nombre de caractères et soit séparé par le même caractère.
  Par exemple : LYO_2012_65510644

- Concernant le protocole de mesure, faire une marque sur l'appareil afin que chaque équipe le pose de la même manière, afin d'éviter les décalages de 180°. Il est aussi envisageable de concevoir un système de visé fixé sur l'appareil pour éviter l'erreur sur la mesure d'orientation de départ ou de définir précisément la manière de la mesurer.
  Concernant les objets laisser sur le sols ou sur les arbres, il est important que les agents les placent hors de vue du scan, car cela dégrade la détection du sol, peut masquer le sols ou la géométrie des arbres. Dans le cas de vestes ou manteaux accrochés aux arbres cela peut même modifier la structure de l'arbre détecté.
- Concernant la fiche terrain, l'information sur l'intensité du vent peut être très pertinente par contre elle doit être établis de manière objective.
- Le taux d'occlutation étant très sensible au sous étage et à la végétation proche de l'appareil, il serait intéressant de définir une échelle pour quantifié la quantité de végétation basse et la présence de sous étage.

Conclusion

Il est important de rappeler que les recommandations sont faits dans le cadre de l'utilisation du jeu de donnée à des fins de validation de logiciel, il se peut très bien qu'elle ne prennent pas en compte les autres utilisations du jeu de données.

Annexe II

Pour plus de lisibilité les includes ont été supprimés, et les headers n'ont pas été inclus dans les annexes, l'intégralité des codes sources peuvent être mis à disposition en par M.Alexandre Piboule du pôle de recherche de l'ONF de Nancy, sur demande.

Code de la fonction d'automatisation du Bach de Computree

```
void MyThread::run()
{

    qDebug()<<"on est dans le thread!!!";


    //efface les étapes à l'écran
    _waitForStepRemoved = true;
    _batch->asyncRemoveAllStep();

    while(_waitForStepRemoved)
        msleep(1000);


    //on parcoure tout les scans du fichiers
    QDir  directory(this->_scansFolderPath);
    int nbFiles = directory.entryInfoList().size();
    for(int i = 0 ; i < nbFiles && !_stop ; i++)
    {

        // lecture d'un fichier script
        QFile file( this->_scriptFilePath);

        //test si le fichier existe l'ouvre est copi le contenu dans script
        if(file.open(QFile::ReadOnly))
        {
            QTextStream stream(&file);
            QString fileText = stream.readAll();
            file.close();
```

```
            //On récupére le chemin du ième fichier
            QString scanTotalPath;
            QString filePath = directory.entryInfoList().at(i).fileName();
            qDebug() << "extension= " <<  filePath.mid(filePath.size()-4,4);
            if (filePath.mid(filePath.size()-4,4) == ".xyb")
            {
                scanTotalPath=getScansFolderPath()+"/"+ filePath;
                qDebug()<<"Le chemin du"<< i <<"eme scan est : "<<scanTotalPath;


                //On modifie le fileText
                fileText=changeScript(this->_scriptFilePath,scanTotalPath);

                //On charche le script
                _waitForScriptLoaded = true;

                _batch->asyncLoadScriptFromString(fileText);

                while(_waitForScriptLoaded)
                    msleep(500);

                if(_batch->execute())
                {
                    while(_batch->isRunning()
                          && !_stop)
                        msleep(1000);

                    if(_stop)
                        _batch->stop();

                    while(_batch->isRunning())
                        msleep(1000);
                }
            }
        }

        //efface les étapes à l'écran
        _waitForStepRemoved = true;
        _batch->asyncRemoveAllStep();

        while(_waitForStepRemoved)
            msleep(1000);
    }
}


QString MyThread::changeScript(QString scriptPath,QString scantotalpath)
{
    //Création du QDomDocument et chargement du script
    QDomDocument script;
    QFile file(scriptPath);
    file.open(QFile::ReadOnly);
    script.setContent(&file);
    file.close();

    //Modification du chemin du scan
    QDomElement scriptElement=script.documentElement();
        //qDebug()<<scriptElement.tagName();
    scriptElement =scriptElement.firstChildElement("Step");
        //qDebug()<<scriptElement.tagName();
    scriptElement =scriptElement.firstChildElement("S1");
        //qDebug()<<scriptElement.tagName();
    scriptElement =scriptElement.firstChildElement("Settings");
        //qDebug()<<scriptElement.tagName();
    scriptElement =scriptElement.firstChildElement("CT_AbstractStepLoadFile");
        //qDebug()<<scriptElement.tagName();
    scriptElement =scriptElement.firstChildElement("FilePath");
        //qDebug()<<scriptElement.tagName();
        qDebug()<<"ancienne valeur du chemin"<<scriptElement.attribute("value");

        scriptElement.setAttribute("value",scantotalpath);
        qDebug()<<"nouvelle valeur du chemin"<<scriptElement.attribute("value");

    //reconvertir en Qstring
     return script.toString();
}
```

Code de l'étape de calcule des indicateurs placettes

```
#define DEF_SearchInScene   "sc"
#define DEF_SearchOutGroup  "g"
#define DEF_SearchOutScene  "sc"
#define DEF_SearchOutResult "r"
#define DEF_SearchInMNT             "mnt"
#define DEF_SearchInMNTResult       "rmnt"

OI_StepCalculatePlotIndicator::OI_StepCalculatePlotIndicator(CT_StepInitializeData &dataInit) :
CT_AbstractStep(dataInit)
{
    _h=2;
    _Rmax=15;
    _azimuthalResolution=0.0359879;
    _zenithalResolution=0.037422;
    _distanceIncrement=0.005;
}

QString OI_StepCalculatePlotIndicator::getStepDescription() const
{
    return tr("Calcul les indicateurs placettes");
}

CT_VirtualAbstractStep* OI_StepCalculatePlotIndicator::createNewInstance(CT_StepInitializeData &dataInit)
{
    // cree une copie de cette etape
    return new OI_StepCalculatePlotIndicator(dataInit);
}

////////////////////// PROTECTED //////////////////

void OI_StepCalculatePlotIndicator::createInResultModelListProtected()
{
    // résultat MNT
    // a besoin en entrée d'un résultat contenant zero ou plusieurs groupes
    CT_InOneOrMoreGroupModel *MNTgroup = new CT_InOneOrMoreGroupModel();
    // le groupe doit contenir un ItemDrawable de type Rester2D
    CT_InStandardItemDrawableModel *MNTitem = new
CT_InStandardItemDrawableModel(CT_Raster2DFloat::staticGetType(),

CT_InAbstractModel::C_ChooseOneIfMultiple,
                                                                    DEF_SearchInMNT,

CT_InStandardItemDrawableModel::F_IsObligatory,
                                                                    "MNT");
    MNTgroup->addItem(MNTitem);
    addInResultModel(new CT_InResultModelGroup(MNTgroup, DEF_SearchInMNTResult, false, "Un résultat avec
un MNT (Raster2D)"));

    CT_InOneOrMoreGroupModel *group = new CT_InOneOrMoreGroupModel();
    CT_InStandardItemDrawableModel *item = new CT_InStandardItemDrawableModel(CT_Scene::staticGetType(),

CT_InStandardItemDrawableModel::C_ChooseOneIfMultiple,
                                                                    DEF_SearchInScene,

CT_InStandardItemDrawableModel::F_IsObligatory,
                                                                    "Scène");
    group->addItem(item);

    // accepte d'etre ajoute si l'etape precedente retourne un resultat contenant seulement UNE scene
    addInResultModel(new CT_InResultModelGroup(group, "r", false,"Un résultat avec une/des scènes"));
}

void OI_StepCalculatePlotIndicator::createPostConfigurationDialog()
{
    CT_StepConfigurableDialog *configDialog = newStandardPostConfigurationDialog();

        configDialog->addDouble(tr("Hauteur"), "m", 0, 1000, 2, _h);
        configDialog->addDouble(tr("Rayon mac de la placette :"), "m", 0, 1000, 2, _Rmax);
        configDialog->addDouble(tr("Résolution azimutal"), "degré", 0, 1000, 5, _azimuthalResolution);
        configDialog->addDouble(tr("Résolution zénithal :"), "degré", 0, 1000, 5, _zenithalResolution);
        configDialog->addDouble(tr("Précision de détection de la frontière de la
placette"),"m",0,100,3,_distanceIncrement);
        configDialog->addFileChoice("Choix du fichier de sortie", CT_FileChoiceButton::OneNewFile,
"Fichier texte (*.*)", _filename);
```

```
}

void OI_StepCalculatePlotIndicator::preProcessCreateOutResultModelListProtected()
{
// Méthode pour compiler du code avant la création de la OutResulModelListProtected
}


void OI_StepCalculatePlotIndicator::createOutResultModelListProtected()
{
    // cette methode est appele apres avoir affiche la fenetre de post-configuration

    CT_OutStandardGroupModel *group = new CT_OutStandardGroupModel(DEF_SearchOutGroup);
    CT_OutStandardItemDrawableModel *item = new CT_OutStandardItemDrawableModel(new CT_Scene(),
DEF_SearchOutScene, "Scène extraite");
    group->addItem(item);

    // genere un resultat avec juste une scene
    addOutResultModel(new CT_OutResultModelGroup(group,  DEF_SearchOutResult, "ExtractedPlot"));

}

void OI_StepCalculatePlotIndicator::compute()


    {
    qDebug()<< "on est dans le compute";

    // récupération du nom de fichier de scan
    QString plotID = "";
    Step* parent = this;
    while (parent->parentStep() != NULL) {parent = parent->parentStep();}

    CT_AbstractStepLoadFile* loadFileStep = dynamic_cast<CT_AbstractStepLoadFile*>(parent);
    if (loadFileStep == NULL) {
        return;
    } else {
        QString plotPath = loadFileStep->getFilePath();
        QFileInfo info(plotPath);
        plotID = info.completeBaseName();
    }


    // Branchement du fichier de sortie
    QFile file(_filename);
    QTextStream stream(&file);

    if (!file.exists())
    {
        file.open(QIODevice::WriteOnly | QIODevice::Text);

        stream << "IDPlacette\t";
        stream << "Nombre points\t";
        stream << "Taux de volume occulté";
        stream << "\n";

        file.close();
    }

        // recupere les resultats d'entree
        // les resultats sont dans l'ordre qu'on les a demande dans la methode
createInResultModelListProtected()
        QList<CT_ResultGroup*> inResultList = getInputResults();

        // récupération du MNT
        CT_ResultGroup *inMNTResult = inResultList.at(0);
        CT_InAbstractItemDrawableModel *inMNTModel = (CT_InAbstractItemDrawableModel*)
getInModel(*inMNTResult->model(), DEF_SearchInMNT);
        inMNTResult->recursiveBeginIterateItems(*inMNTModel);
        CT_AbstractItemGroup *groupWithMNT = inMNTResult->recursiveNextItem();
        groupWithMNT->beginIterateItems(*inMNTModel);
        CT_Raster2DFloat *mnt = (CT_Raster2DFloat*)groupWithMNT->nextItem();

        if (mnt != NULL)
            {
            CT_ResultGroup *inResult = inResultList.at(1);
            CT_InAbstractItemDrawableModel *inSceneModel = (CT_InAbstractItemDrawableModel*)
getInModel(*inResult->model(), DEF_SearchInScene);
            // on demande si on peut commencer à parcourir les groupes contenant une scène
```

```
            if(inResult->recursiveBeginIterateItems(*inSceneModel))
                {
                CT_AbstractItemGroup *groupWithScene = inResult->recursiveNextItem();

                if(groupWithScene->beginIterateItems(*((CT_InAbstractItemDrawableModel*)inSceneModel),
false))
                    {
                    const CT_Scene *in_scene = (CT_Scene*)groupWithScene->nextItem();

                    const CT_AbstractPointCloud *pointCloud = in_scene->getPointCloud();
                    const CT_AbstractPointCloudIndex *pointCloudIndex = in_scene->getPointCloudIndex();
                    int n_points = pointCloudIndex->indexSize();

                    //initialisation des variable
                    float vtot;
                    float coordX; // coordonnée X du ième point
                    float coordY;   //coordonnée Y du i ème point
                    float vocculte;
                    float d; //distance xy du point au centre, on ne prend pas en compte le z
                    double R; // R est la distance du centre et de l'intersection avec la frontiere de la
placette
                              // est égale d si l'intersection est en dehors du MNT car dans ce cas la
                             // le point n'occulte pas de volume dans la placette.

                    quint64 numberPointInPlot;

                    //Calcul de Vtot
                    vtot= _h*_Rmax*_Rmax*M_PI;
                    vocculte = 0;
                    numberPointInPlot =0;
                    _numberPointInAera = 0;

                    // Extraction des points de la placette
                    quint64 i=0;
                    while((i<n_points)
                            && (!isStopped()))
                        {
                        const int &index = (*pointCloudIndex)[i];
                        const CT_Point &point = (*pointCloud)[index];
                        coordX =point.x;
                        coordY= point.y;
                        d= sqrt(coordX*coordX+coordY*coordY);
                         if(d<_Rmax)
                            {
                            R=calculateR( point,mnt, _distanceIncrement);
                            // On compte le nombre de point dans la fonction calculate R
                            //On convertie les résolutions zénital et azimutal de degrer en radian

vocculte=vocculte+2*M_PI/360*_zenithalResolution*2*M_PI/360*_azimuthalResolution*(R*R*R-d*d*d)/3;
                            }

                        // progres de 0  100
                        setProgress((i*100)/n_points);
                        ++i;
                    }

                    _rateOccultedVolume=vocculte/vtot;

            //Résultat
                    qDebug() << "Le nombre de points dans la scene : " << n_points;
                    qDebug()<< "Le nombre de point situé dans la frontière : " <<_numberPointInAera;
                    qDebug()<< "Le volume total égal : "<< vtot;
                    qDebug()<< "Le Volume Occulté est : "<< vocculte;
                    qDebug()<< "le taux de volume occulté est : " << _rateOccultedVolume;


             //écriture des indicateurs
                    if (!file.open(QIODevice::Append | QIODevice::Text))
                    {
                        qDebug() << "Indicateurs calculés mais impossible de les exporter !";
                        return;
                    }

                    stream << plotID << "\t";
                    stream << _numberPointInAera<< "\t";
                    stream << _rateOccultedVolume;
                    stream << "\n";
                    file.close();
```

```
                }
            }
        }

    }

double OI_StepCalculatePlotIndicator::calculateR(CT_Point point,CT_Raster2DFloat *MNT,double increment )
{
    float hauteur1,hauteur2;
    float X,Y,Z;// les coordonnée en X et Y du point d'intersection entre la frontière en la droite
passant par le centre et le point.
    float resX,resY,resZ,R;                 // le pas duquel on avance a chaque boucle pour détecter la
frontière.

    //initialisation
    X=point.x;
    Y=point.y;
    Z=point.z;
    R=sqrt(X*X+Y*Y+Z*Z);
    resX= increment*X/R;
    resY=increment*Y/R;
    resZ=increment*Z/R;
    hauteur1=increment;
    hauteur2=increment;
    bool PointInAera(false);

    do
        {
        //On avance d'un pas sur le rayon partant du centre passant par le point
        X=X+resX;
        Y=Y+resY;
        Z=Z+resZ;
        R=sqrt(X*X+Y*Y+Z*Z);
        float na;
        float zMNT = MNT->valueAtXY(X,Y, na);
        if (zMNT != na) {
            hauteur1 = Z - zMNT;
            hauteur2 = _h-hauteur1;
        } else {
            //récupérer la valeur de na
            //hauteur = LONG MIN;
        }

        if(hauteur1>0 and hauteur2>0)
           { PointInAera = true;}

    } while (hauteur1>0 and hauteur2>0 and R<_Rmax);

    if( PointInAera==true)
    {
    _numberPointInAera=_numberPointInAera+1;
    }

  return R;
}
```

Code de l'étape de calcule des indicateurs arbres

```
#define DEF_SearchInCylinder   "cyl"
#define DEF_SearchInRefPoint   "rf"
#define DEF SearchInSectionGroup   "sg"
#define DEF SearchInPointClusterGroupe "pcg"
#define DEF SearchInPointCluster "pc"
#define DEF_SearchInCircle "cir"
#define DEF_SearchInRefPoint2 "rf2"
#define DEF_SearchInResult  "r"
#define DEF SearchOutResult "r"


OI_StepCalculateTreeIndicator::OI_StepCalculateTreeIndicator(CT_StepInitializeData &dataInit) :
CT_AbstractStep(dataInit)
{
    _hmin=0.3;
    _hmax = 2.3;
    _azimuthalResolution = 0.0359879;
    _zenithalResolution = 0.037422;
    _layerHeight = 0.1;
    _minPointByLayer = 5;
    _filename = "";
}

QString OI_StepCalculateTreeIndicator::getStepDescription() const
{
    return tr("Calcul les indicateurs pour chacun des arbres numériques.");
}

CT_VirtualAbstractStep* OI_StepCalculateTreeIndicator::createNewInstance(CT_StepInitializeData &dataInit)
{
    // cree une copie de cette etape
    return new OI_StepCalculateTreeIndicator(dataInit);
}

void OI_StepCalculateTreeIndicator::createInResultModelListProtected()
{
    CT_InZeroOrMoreGroupModel *group = new CT_InZeroOrMoreGroupModel();
    CT_InStandardGroupModel* sectionGroup = new CT_InStandardGroupModel(DEF_SearchInSectionGroup,
"Section","",CT_InAbstractModel::C_ChooseOneIfMultiple, CT_InAbstractGroupModel::FG_IsObligatory);

CT_InStandardItemDrawableModel *circle =
newCT_InStandardItemDrawableModel(CT_Circle::staticGetType(),CT_InStandardItemDrawableModel::C_ChooseOneIf
Multiple, DEF_SearchInCircle, CT_InStandardItemDrawableModel::F_IsObligatory, "cercles ajuster à 1m30");

CT_InStandardItemDrawableModel *refpoint2 =
newCT_InStandardItemDrawableModel(CT_ReferencePoint::staticGetType(),CT_InStandardItemDrawableModel::C_Cho
oseOneIfMultiple, DEF_SearchInRefPoint2, CT_InStandardItemDrawableModel::F_IsObligatory, Coordonnée MNT");

CT_InStandardGroupModel* pointClusterGroup = new CT_InStandardGroupModel(DEF_SearchInPointClusterGroupe,
"Groupe cluster","",CT_InAbstractModel::C_ChooseOneIfMultiple, CT_InAbstractGroupModel::FG_IsObligatory);

CT_InStandardItemDrawableModel *cyl = new
CT_InStandardItemDrawableModel(CT_Cylinder::staticGetType(),CT_InStandardItemDrawableModel::C_ChooseOneIfM
ultiple, DEF_SearchInCylinder, CT_InStandardItemDrawableModel::F_IsObligatory, "Cylindre");


CT_InStandardItemDrawableModel *pointCluster = new
CT_InStandardItemDrawableModel(CT_PointCluster::staticGetType(),CT_InStandardItemDrawableModel::C_ChooseOn
eIfMultiple, DEF_SearchInPointCluster, CT_InStandardItemDrawableModel::F_IsObligatory, Point Cluster");

    CT_InStandardItemDrawableModel *refPoint = new
CT_InStandardItemDrawableModel(CT_ReferencePoint::staticGetType(),CT_InStandardItemDrawableModel::C_Choose
OneIfMultiple, DEF_SearchInRefPoint, CT_InStandardItemDrawableModel::F_IsObligatory,"barycentre des points
cluster");


    group->addGroup(sectionGroup);
    sectionGroup->addItem(circle);
    sectionGroup->addItem(refpoint2);
    sectionGroup->addGroup(pointClusterGroup);
    pointClusterGroup->addItem(cyl);
    pointClusterGroup->addItem(pointCluster);
    pointClusterGroup->addItem(refPoint);
```

```
    // accepte d'etre ajoute si l'etape precedente retourne un resultat contenant seulement UNE scene
    addInResultModel(new CT_InResultModelGroupToCopy(group, DEF_SearchInResult, false, "Un résultat avec
des sections contenant des cylindres"));
}

void OI_StepCalculateTreeIndicator::createOutResultModelListProtected()
{
    // on récupère le résultat modèle d'entrée à copier défini dans "createInResultModelListProtected()"
    CT_InResultModelGroupToCopy *intResModelToCopy =
(CT_InResultModelGroupToCopy*)getInResultModel(DEF_SearchInResult);

    // on récupère toutes les possibilités que l'utilisateur à défini
    QList<CT_OutResultModelGroupToCopyPossibility*> copyList = intResModelToCopy-
>getOutResultModelGroupsSelectedToCopy();
    QListIterator<CT_OutResultModelGroupToCopyPossibility*> it(copyList);

    // pour chaque possibilité à copier/modifier (il n'y en a qu'une ici puisque 1 modèle = 1 possibilité
par défaut)
    if(it.hasNext())
    {
        CT_OutResultModelGroupToCopyPossibility *outResModelToCopy = it.next();

        // on crée un nouveau résultat de sortie contenant le résultat modèle modifié
        addOutResultModel(new CT_OutResultModelGroupCopy(DEF_SearchOutResult, outResModelToCopy));
    }
}

void OI_StepCalculateTreeIndicator::createPostConfigurationDialog()
{
    CT_StepConfigurableDialog *configDialog = newStandardPostConfigurationDialog();

        configDialog->addDouble(tr("Hmin :"), "m", 0, 1000, 2, _hmin);
        configDialog->addDouble(tr("Hmax :"), "m", 0, 1000, 2, _hmax);
        configDialog->addDouble(tr("Résolution azimutal :"), "degré", 0, 360, 5, _azimuthalResolution);
        configDialog->addDouble(tr("Résolution zenithal :"), "degré", 0, 360, 5, _zenithalResolution);
        configDialog->addDouble(tr("Hauteur des tranches pour discrétiser les sections :"), "m", 0, 100,
2, _layerHeight);
        configDialog->addDouble(tr("Nombre de point min par Tranche :"), "nb points", 0, 100000, 0,
_minPointByLayer);
        configDialog->addFileChoice("Choix du fichier de sortie", CT_FileChoiceButton::OneNewFile,
"Fichier texte (*.*)", _filename);
}

void OI_StepCalculateTreeIndicator::compute()
{
    qDebug() << "On est dans le compute";
    // on récupère le résultat copié
    CT_ResultGroup *outRes = getOutResultList().first();

    /********* RECHERCHE DES MODELES D'ENTREE ************/

    // on récupère le modèle d'entrée qu'on avait ajouté
    CT_InResultModelGroupToCopy *intResModel =
(CT_InResultModelGroupToCopy*)getInResultModel(DEF_SearchInResult);
    CT_OutResultModelGroup *outModelForSearchInModel = (CT_OutResultModelGroup*) intResModel-
>getOutResultModelForSearchInModel().first();
    // et qu'on utilise ici pour rechercher le modèle d'entrée du nuage de point
    CT_InAbstractItemDrawableModel *inCircleModel =
(CT_InAbstractItemDrawableModel*)getInModel(*outModelForSearchInModel, DEF_SearchInCircle);
    CT_InAbstractItemDrawableModel *inRefPoint2Model =
(CT_InAbstractItemDrawableModel*)getInModel(*outModelForSearchInModel, DEF_SearchInRefPoint2);
    CT_InAbstractItemDrawableModel *inCylinderModel =
(CT_InAbstractItemDrawableModel*)getInModel(*outModelForSearchInModel, DEF_SearchInCylinder);
    CT_InAbstractItemDrawableModel *inRefPointModel =
(CT_InAbstractItemDrawableModel*)getInModel(*outModelForSearchInModel, DEF_SearchInRefPoint);
    CT_InAbstractGroupModel *inSectionModel =
(CT_InAbstractGroupModel*)getInModel(*outModelForSearchInModel, DEF_SearchInSectionGroup);
    CT_InAbstractGroupModel *inPointClusterGroupModel =
(CT_InAbstractGroupModel*)getInModel(*outModelForSearchInModel, DEF_SearchInPointClusterGroupe);
    CT_InAbstractItemDrawableModel *inPointClusterModel =
(CT_InAbstractItemDrawableModel*)getInModel(*outModelForSearchInModel, DEF_SearchInPointCluster);

    /************************************************/

    // récupération du nom de fichier de scan
    QString plotID = "";
    Step* parent = this;
    while (parent->parentStep() != NULL) {parent = parent->parentStep();}
```

```
CT_AbstractStepLoadFile* loadFileStep = dynamic_cast<CT_AbstractStepLoadFile*>(parent);
if (loadFileStep == NULL) {
    return;
} else {
    QString plotPath = loadFileStep->getFilePath();
    QFileInfo info(plotPath);
    plotID = info.completeBaseName();
}


// Branchement du fichier de sortie
QFile file(_filename);
QTextStream stream(&file);

if (!file.exists())
{
    file.open(QIODevice::WriteOnly | QIODevice::Text);

    stream << "IDPlacette\t";
    stream << "IDArbreNum\t";
    stream << "DiametreEstime\t";
    stream << "DiametreArbre\t";
    stream << "DistanceArbre\t";
    stream << "AzimutArbreRAD\t";
    stream << "NbPoint\t";
    stream << "NbCylindres\t";
    stream << "NbClusters\t";
    stream << "DBT\t";
    stream << "OfOverestimatedC13\t";
    stream << "ShapePC degré/m\t";
    stream << "ShapeVC degré/m\t";
    stream << "NbArc\t";
    stream << "RateArc\t";
    stream << "QuartInfArc\t";
    stream << "MedArc\t";
    stream << "QuartSupArc\t";
    stream << "Zmin\t";
    stream << "Zmax\t";
    stream << "AzimuteMaxMinRAD";
    stream << "\n";

    file.close();
}

//initialisation
_numberNumTreeDetected=0;

if(outRes->recursiveBeginIterateGroups(*inSectionModel))
{
    CT_AbstractItemGroup *section;
    int idSection = -1;

    while((section = outRes->recursiveNextGroup()) != NULL && (!isStopped()))
    {
        //réinitialisation par arbre
        _numberPointCluster=0;
        _boolEstimatedRadius=false;

        _indNbPoint = 0;
        _indDBT = 0;
        _indOfOverestimatedC13 = false;
        _indShapePC = 0;
        _indShapeVC = 0;
        _indNbArc =  0;
        _indRateArc = 0;
        _indMedArc = 0;
        _indQuartSupArc = 0;
        _indQuartInfArc = 0;

        ++_numberNumTreeDetected;
        double moduleXY=0;
        double azimute=0;
        double zmin = std::numeric_limits<double>::max();
        double zmax = std::numeric_limits<double>::min();
        double azimutmin = 4*M_PI;
        double azimutmax = -4*M_PI;
        section->beginIterateItems(inCircleModel);
        if(section->containsItemDrawable(*inCircleModel)==true)
```

```
                {
                    _boolEstimatedRadius=true;
                    CT_Circle* circle = (CT_Circle*) section->findFirstItem(*inCircleModel,
CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);
                    _RadiusOfNumTree=circle->getRadius();
                    _dprOfNumTree=sqrt(circle->getCenterX()*circle->getCenterX()+circle-
>getCenterY()*circle->getCenterY());
                    _azimutOfNumTree = acos(circle->getCenterY()/_dprOfNumTree);
                    if (asin(circle->getCenterX()/_dprOfNumTree) < 0) {_azimutOfNumTree = (2*M_PI -
_azimutOfNumTree);}

                    /*qDebug()<<"";
                            qDebug()<<"";
                            qDebug()<<"L'arbre à un rayon a 1m30 estimé de "<< RadiusOfNumTree;
                            qDebug()<<"l'arbre à dpr de "<<_dprOfNumTree;*/
                }

            section->beginIterateItems(inRefPoint2Model);
            if(section->containsItemDrawable(*inRefPoint2Model)==true)
                {
                CT_ReferencePoint* refPoint = (CT_ReferencePoint*) section-
>findFirstItem(*inRefPoint2Model, CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);
                _zmntOfNumTree=refPoint->getCenterZ();
                idSection = refPoint->refId();

                if(_boolEstimatedRadius==false)
                        {
                            _dprOfNumTree=sqrt(refPoint->getCenterX()*refPoint->getCenterX()+refPoint-
>getCenterY()*refPoint->getCenterY());
                            _azimutOfNumTree = acos(refPoint->getCenterY()/_dprOfNumTree);
                            if (asin(refPoint->getCenterX()/_dprOfNumTree) < 0) {_azimutOfNumTree =
(2*M_PI - _azimutOfNumTree);}

                            /*qDebug()<<"";
                            qDebug()<<"";
                            qDebug()<<"L'arbre n'a pas de rayon estimé à 1m30";
                            qDebug()<<"L'arbre a un dpr de "<<_dprOfNumTree;*/
                        }
                 }
                //qDebug()<<"La hauteur du mnt au pied de l'arbre est : "<< zmntOfNumTree;


//Boucle de recherche de zmin zmax et limites en azimut

            section->beginIterateGroups(inPointClusterGroupModel);
            CT_AbstractItemGroup* group;
            while ((group = section->nextGroup()) != NULL && (!isStopped()))
            {
                bool pointsInZone = false;
                const CT_PointCluster* pointCluster = (CT_PointCluster*) group-
>findFirstItem(*inPointClusterModel, CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);
                if (pointCluster!=NULL)
                {
                    const CT_AbstractPointCloud* pointCloud = pointCluster->getPointCloud();
                    const CT_AbstractPointCloudIndex *pointCloudIndex = pointCluster-
>getPointCloudIndex();
                    int i = 0;
                    while((i<pointCloudIndex->indexSize()) && (!isStopped()))
                    {

                        const int &index = (*pointCloudIndex)[i];
                        const CT_Point &point = (*pointCloud)[index];
                        //recherche de zmin et zmax
                        if((point.z>_hmin+_zmntOfNumTree) && (point.z<_hmax+_zmntOfNumTree))
                        {
                            _indNbPoint++;
                            pointsInZone = true;
                            if(point.z<zmin){zmin=point.z;}
                            if(point.z>zmax){zmax=point.z;}

                            float distance = sqrt(point.x*point.x + point.y*point.y);
                            azimute = acos(point.y/distance);
                            if (asin(point.x/distance) < 0) {azimute = (2*M_PI - azimute);}

                            if(azimute<azimutmin){azimutmin=azimute;}
                            if(azimute>azimutmax){azimutmax=azimute;}
                        }
```

```
                        ++i;
                    }
                    if (pointsInZone) {++_numberPointCluster;}

                }
            }

            //Important poue les cas ou l'arbre est sur l'axe y la recherche du min et du max bug il faud
chercher le max dans et le min localement
            if(azimutmin<=M_PI/2 && azimutmax>3*M_PI/2)
                {
                    azimutmin = 4*M_PI;
                    azimutmax = -4*M_PI;
                    section->beginIterateGroups(inPointClusterGroupModel);
                    CT_AbstractItemGroup* group;
                    while ((group = section->nextGroup()) != NULL && (!isStopped()))
                    {
                        bool pointsInZone = false;
                        const CT_PointCluster* pointCluster = (CT_PointCluster*) group-
>findFirstItem(*inPointClusterModel, CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);
                        if (pointCluster!=NULL)
                            {
                            const CT_AbstractPointCloud* pointCloud = pointCluster->getPointCloud();
                            const CT_AbstractPointCloudIndex *pointCloudIndex = pointCluster-
>getPointCloudIndex();
                            int i = 0;
                            while((i<pointCloudIndex->indexSize()) && (!isStopped()))
                            {

                                const int &index = (*pointCloudIndex)[i];
                                const CT_Point &point = (*pointCloud)[index];
                                //recherche de zmin et zmax
                                if((point.z>_hmin+_zmntOfNumTree) && (point.z<_hmax+_zmntOfNumTree))
                                {
                                    _indNbPoint++;
                                    float distance = sqrt(point.x*point.x + point.y*point.y);
                                    azimute = acos(point.y/distance);
                                    if (asin(point.x/distance) < 0) {azimute = (2*M_PI - azimute);}

                                    if(azimute<M_PI && azimute>azimutmax){azimutmax=azimute;}
                                    if(azimute>M_PI && azimute<azimutmin){azimutmin=azimute;}
                                }

                                ++i;
                            }

                        }
                    }
                    qDebug()<<"azimute max locas"<<azimutmax*180/M_PI;
                    qDebug()<<"azimute min local"<<azimutmin*180/M_PI;
                    azimutmax=azimutmax+2*M_PI-azimutmin;
                    azimutmin=0;

                }


//Boucle pour remplir le tableNumberPointByTranche
            //initialisation
            QVector<int> tableNumberPointByTranche;
            QVector<double> tableHeightTranche;
            int numberPointByTranche;
            double h = zmin;
            while(h<=zmax)
            {
                section->beginIterateGroups(inPointClusterGroupModel);
                numberPointByTranche=0;

                while ((group = section->nextGroup()) != NULL && (!isStopped()))
                {
                    const CT_PointCluster* pointCluster = (CT_PointCluster*) group-
>findFirstItem(*inPointClusterModel, CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);
                    if (pointCluster!=NULL)
                        {
                        const CT_AbstractPointCloud* pointCloud = pointCluster->getPointCloud();
                        const CT_AbstractPointCloudIndex *pointCloudIndex = pointCluster-
>getPointCloudIndex();
                        //boucle parcourant chaque point de chacque PointCluster
                        int i=0;
                        while((i<pointCloudIndex->indexSize())
```

```
                                && (!isStopped()))
                            {
                            const int &index = (*pointCloudIndex)[i];
                            const CT_Point &point = (*pointCloud)[index];

                            if(point.z>h and point.z<=h+_layerHeight)
                                {
                                 ++numberPointByTranche;
                                 }
                             i++;
                             }
                        }
                }
                    if(numberPointByTranche>ceil(_minPointByLayer))
                        {
                        tableNumberPointByTranche.push_back(numberPointByTranche);
                        tableHeightTranche.push_back(h-_zmntOfNumTree);//on décale la hauteur des section
pour que les coordonnées sont en fonction du mnt
                        }
                    h=h+_layerHeight;
                    numberPointByTranche=0;
            }


              //Initialisation
              CT_AbstractItemGroup* group2;
              QVector<QVector3D> cylinderVectorDirector;
              QVector<double> heightOfCylinder;
              section->beginIterateGroups(inPointClusterGroupModel);

//Boucle récupérant les vecteur directeur de chaque cylindre ainsi que la hauteur
              while ((group2 = section->nextGroup())!=NULL && (!isStopped()))
              {
                  CT_Cylinder* cylinder = (CT_Cylinder*) group2->findFirstItem(*inCylinderModel,
CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);

                  if (cylinder!=NULL)
                  {
                      if(cylinder->getCenterZ()<zmax and cylinder->getCenterZ()>zmin)
                      {
                          cylinderVectorDirector.push_back(cylinder->getDirection());
                          heightOfCylinder.push_back(cylinder->getCenterZ());
                      }

                  }

              }

              int i,j;
              i=0;
              j=0;
              //boucles pour classer par ordre croissant les cylindres
              if(cylinderVectorDirector.size()>=2)
              {
                  int tablesize(heightOfCylinder.size());
                  while (i<tablesize-3)
                    {
                    for (j=i; j<tablesize-2;j++)
                        {
                        if(heightOfCylinder[j]>heightOfCylinder[j+1])
                            {
                            double heightCopy;
                            QVector3D vectorCopy;
                            heightCopy=heightOfCylinder[j];
                            vectorCopy=cylinderVectorDirector[j];
                            heightOfCylinder[j]=heightOfCylinder[j+1];
                            cylinderVectorDirector[j]=cylinderVectorDirector[j+1];
                            heightOfCylinder[j+1]=heightCopy;
                            cylinderVectorDirector[j+1]=vectorCopy;
                            }
                        }
                        i++;
                    }
                  }

//boucle de remplissage de refPointVector
                  //Initialisation
                  CT_AbstractItemGroup* group3;
                  QVector<QVector3D> refPointVector;
```

```
                section->beginIterateGroups(inPointClusterGroupModel);

                //Boucle récupérent les coordonnée des barycentres
                while ((group3 = section->nextGroup())!=NULL && (!isStopped()))
                {
                    CT_ReferencePoint* refPoint = (CT_ReferencePoint*) group3-
>findFirstItem(*inRefPointModel, CT_AbstractItemGroup::S_Backup | CT_AbstractItemGroup::S_New);

                    if (refPoint!=NULL)
                    {
                        if(refPoint->getCenterZ()>zmin and refPoint->getCenterZ()<zmax)
                        {
                            QVector3D vector(refPoint->getCenterX(),refPoint->getCenterY(),refPoint-
>getCenterZ());
                            refPointVector.push_back(vector);
                        }
                    }

                }
                i=0;
                j=0;
                //boucles pour classer par ordre croissant les cylindres
                if(refPointVector.size()>=3)
                    {
                    int tablesize(refPointVector.size());
                    while (i<tablesize-3)
                        {
                        for (j=i; j<tablesize-2;j++)
                            {
                            if(refPointVector[j].z()>refPointVector[j+1].z())
                                {
                                QVector3D vectorCopy;
                                vectorCopy=refPointVector[j];
                                refPointVector[j]=refPointVector[j+1];
                                refPointVector[j+1]=vectorCopy;
                                }
                            }
                        i++;
                        }
                    }



//Calcule des Resultats

             //calcul de l'esimation du nombre de point avec zmin zmax azimutmin azimute max
             double teta2,alpha1,alpha2,alphatot;
             double trancheVertical3(0);
             double trancheHorizontal3(0);

             alpha1=acos(_dprOfNumTree/sqrt(pow(_dprOfNumTree,2)+pow(zmax,2)));
             alpha2=acos(_dprOfNumTree/sqrt(pow(_dprOfNumTree,2)+pow(zmin,2)));
             alphatot=fabs(alpha1*(zmax/fabs(zmax))-alpha2*(zmin/fabs(zmin)));
             teta2=azimutmax-azimutmin;
             qDebug()<<"L'arbre n°"<<_numberNumTreeDetected<<" azimut min"<<azimutmin*360/(2*M_PI)<<"
azimut max."<<azimutmax*180/M_PI;
             qDebug()<<"L'arbre n°"<<_numberNumTreeDetected<<" angle tot"<<teta2*360/(2*M_PI);
             qDebug()<<"";
             trancheHorizontal3=alphatot/(_zenithalResolution*M_PI/180);
             trancheVertical3=teta2/(_azimuthalResolution*M_PI/180);

             _indDBT=_indNbPoint/(trancheVertical3*trancheHorizontal3);


            //calcul de l'esimation du nombre de point avec zmin zmax azimutmin azimute max
            if(_boolEstimatedRadius==true)
            {
                 double r13(_RadiusOfNumTree);
                 double trancheVertical2(0);
                 double trancheHorizontal2(0);
                 double teta,alpha1,alpha2,alphatot;

                 alpha1=acos(_dprOfNumTree/sqrt(pow(_dprOfNumTree,2)+pow(zmax,2)));
                 alpha2=acos(_dprOfNumTree/sqrt(pow(_dprOfNumTree,2)+pow(zmin,2)));
                 alphatot=fabs(alpha1*(zmax/fabs(zmax))-alpha2*(zmin/fabs(zmin)));
                 teta=acos(_dprOfNumTree/sqrt(pow(_dprOfNumTree,2)+pow(r13,2)));
                 trancheHorizontal2=alphatot/(_zenithalResolution*M_PI/180);
                 trancheVertical2=2*teta/(_azimuthalResolution*M_PI/180);
```

```
                if(_indNbPoint/(trancheVertical2*trancheHorizontal2)>1)
                {
                    _indOfOverestimatedC13=true;
                }
            }

            //Calcul des indices sur la répartition des arc de cercle sur une section
            _indNbArc=tableHeightTranche.size();
            _indRateArc=_indNbArc/((zmax-zmin)/_layerHeight);
            if(_indNbArc>3)
            {
            _indMedArc= tableHeightTranche[ceil(_indNbArc/2)];//ravaler dan sun repère en fonction du mnt
            _indQuartSupArc=tableHeightTranche[ceil(_indNbArc*3/4)];
            _indQuartInfArc=tableHeightTranche[ceil(_indNbArc/4)];
            }


             //Calcul de l'indice de forme du squellette depuis les cylindres
             double angleCumule(0);
             double angletot(0);
             if(cylinderVectorDirector.size()>=2)
                 {

angletot=360/(2*M_PI)*fabs(acos(QVector3D::dotProduct(cylinderVectorDirector[0],cylinderVectorDirector[cyl
inderVectorDirector.size()-
1])/(cylinderVectorDirector[0].length()*cylinderVectorDirector[cylinderVectorDirector.size()-
1].length())));
                 for(int k=0;k<cylinderVectorDirector.size()-1;k++)
                     {

angleCumule=angleCumule+360/(2*M_PI)*fabs(acos(QVector3D::dotProduct(cylinderVectorDirector[k],
cylinderVectorDirector[k+1] )/(cylinderVectorDirector[k].length()*cylinderVectorDirector[k+1].length())));

                     }
                 _indShapeVC=(angleCumule-angletot)/fabs(zmax-zmin);
                 }
             //Calculer de l'indice de forme du squelette depuis les point refecrence
             angleCumule=0;
             angletot=0;
             if(refPointVector.size()>=3)
                 {
                 QVector3D firstVector;
                 QVector3D finalVector;

                 firstVector =refPointVector[1]-refPointVector[0];
                 finalVector=refPointVector[refPointVector.size()-1]-refPointVector[refPointVector.size()-
2];
                 angletot=360/(2*M_PI)*fabs(acos(QVector3D::dotProduct(firstVector, finalVector
)/(firstVector.length()*finalVector.length())));
                 for(int k=0;k<refPointVector.size()-2;k++)
                 {

angleCumule=angleCumule+360/(2*M_PI)*fabs(acos(QVector3D::dotProduct(refPointVector[k+2]-
refPointVector[k+1], refPointVector[k+1]-refPointVector[k] )/((refPointVector[k+2]-
refPointVector[k+1]).length()*(refPointVector[k+1]-refPointVector[k]).length())));
                 }
                 _indShapePC=(angleCumule-angletot)/fabs(zmax-zmin);

                 }

//Resultat

             qDebug()<<"Les indices de l'arbre n°"<<_numberNumTreeDetected<<"sont :";
             qDebug()<<"L'indice de nombre de point par arbre est de : "<< _indNbPoint;
             qDebug()<<"L'indice de densité de point par arbre est de : "<< _indDBT;

             if(_indOfOverestimatedC13==true)
                 {
                 qDebug()<<"Le diametre estimé est sousestimé!";
                 }
             else
                 {
                 qDebug()<<"Le diametre estimé ne semble pas sousestimé au vu de la densité de point!";
                 }

             if(refPointVector.size()>=3)
                 {
```

```
                qDebug()<<"L'indice de forme du squelette des barycentres des pointCluster est de : "<<
_indShapePC;
                }
            else
                {
                qDebug()<<"L'indice de forme du squelette des barycentres des pointCluster n'est pas
calculabe!";
                }

            if(cylinderVectorDirector.size()>=2)
                {
                qDebug()<<"L'indice de forme du squelette des  cylindres ajustés sur les cylindres : "<<
_indShapeVC;
                }
                else
                {
                qDebug()<<"L'indice de forme du squelette des  cylindres ajustés sur cylindre n'est pas
calculable!";
                }

            qDebug()<<"L'indice du nombre d'arcs de cercle de la section est de : "<< _indNbArc;
            qDebug()<<"Le point le plus haut est à une hauteur de : "<< zmax-_zmntOfNumTree;
            if(_indNbArc>3)
            {
                qDebug()<<"Le quartile supérieur de la répartition des arcs de cercles est de : "<<
_indQuartSupArc;
                qDebug()<<"La hauteur médian de répartition des arcs de cercles est de : "<< _indMedArc;
                qDebug()<<"Le quartile inférieur de la répartition des arcs de cercles est de : "<<
_indQuartInfArc;
            }
            else
            {
                qDebug()<<"Il y a moins de 4 arc de cercle la médian et quartil non calculable!";
            }
            qDebug()<<"Le point le plus bas est à une hauteur de : "<< zmin-_zmntOfNumTree;
            qDebug()<<"L'indice du nombre d'arcs sur le nombre d'arc max est de : "<< _indRateArc;
            qDebug()<<"";
            qDebug()<<"";


            // Export des données de la section
            if (!file.open(QIODevice::Append | QIODevice::Text))
            {
                qDebug() << "Indicateurs calculés mais impossible de les exporter !";
                return;
            }

            stream << plotID << "\t";
            stream << idSection << "\t";
            stream << (_boolEstimatedRadius?"o":"n") << "\t";
            stream << (_boolEstimatedRadius?(_RadiusOfNumTree*2):-1) << "\t";
            stream << _dprOfNumTree << "\t";
            stream << _azimutOfNumTree << "\t";
            stream << _indNbPoint << "\t";
            stream << cylinderVectorDirector.size() << "\t";
            stream << _numberPointCluster << "\t";
            stream << _indDBT << "\t";
            stream << (_indOfOverestimatedC13?"o":"n") << "\t";
            stream << _indShapePC << "\t";
            stream << _indShapeVC << "\t";
            stream << _indNbArc << "\t";
            stream << _indRateArc << "\t";
            stream << _indQuartInfArc << "\t";
            stream << _indMedArc << "\t";
            stream << _indQuartSupArc << "\t";
            stream << (zmin - _zmntOfNumTree) << "\t";
            stream << (zmax - _zmntOfNumTree) << "\t";
            stream << (azimutmax-azimutmin);
            stream << "\n";
            file.close();

        }
    }


}
```