

# Rapport de stage ingénieur

Construction de cartes 3D forestières  
par vision embarquée sur un drone

Sous la supervision de M. Pascal VASSEUR  
[pascal.vasseur@u-picardie.fr](mailto:pascal.vasseur@u-picardie.fr)

Théo LARCHER

INSA - Architecture des Systèmes d'Information - 5<sup>ème</sup> année  
Université de Rouen - Master Sciences des données - M2



1er mars - 31 juillet 2021

# Table des matières

<b>Remerciements</b>	<b>1</b>
<b>I Présentation</b>	<b>2</b>
<b>1 Organisme d'accueil</b>	<b>2</b>
1.1 Généralités . . . . .	2
1.2 Activité . . . . .	3
1.3 Équipes . . . . .	3
1.4 Cadre de travail . . . . .	4
1.5 Contraintes sanitaires . . . . .	4
<b>2 Présentation du sujet de stage</b>	<b>5</b>
2.1 Contexte du projet . . . . .	5
2.2 Sujet de stage . . . . .	5
<b>II Travail effectué</b>	<b>6</b>
<b>1 Cahier des charges</b>	<b>6</b>
1.1 Objectifs du projet . . . . .	6
1.2 Cible du projet . . . . .	6
1.3 Besoins et contraintes liés au projet . . . . .	6
1.4 Périmètre du projet . . . . .	6
1.5 Délais . . . . .	7
1.6 Méthodologie de gestion de projet . . . . .	7
<b>2 Installation et reprise des travaux précédents</b>	<b>7</b>
2.1 Matériel physique ( <i>Hardware</i> ) . . . . .	7
2.2 Matériel virtuel & logiciels ( <i>Software</i> ) . . . . .	8
2.3 Travaux de Dao . . . . .	9
2.4 Reprise des travaux . . . . .	11
2.4.1 Algorithmes . . . . .	11
2.4.2 Données et UnrealCV . . . . .	13
2.5 <i>Datasets</i> et environnements UE4 . . . . .	14
2.5.1 Choix des données . . . . .	14
2.5.2 Comparaison des mondes UE4 . . . . .	15
2.6 Ralentissements au démarrage du projet . . . . .	16
<b>3 Développement</b>	<b>18</b>
3.1 Synchronisation des caméras . . . . .	18
3.1.1 Astuces . . . . .	18
3.1.2 <i>Generator-locked</i> . . . . .	18
3.1.3 Post-traitement . . . . .	19
3.2 Simulateurs et captures . . . . .	20
3.2.1 Unreal Engine 4 Editor . . . . .	20
3.2.2 AirSim . . . . .	20
3.2.3 Création des données . . . . .	21
3.2.4 Projection equirectangulaire . . . . .	24
3.3 Segmentation sémantique . . . . .	28
3.3.1 Monde label . . . . .	28
3.3.2 AdapNet++ . . . . .	28
3.4 Estimation de profondeur . . . . .	30
3.4.1 360SD-Net . . . . .	30
3.5 Création de cartes . . . . .	31

3.5.1	Vérité terrain . . . . .	31
3.5.2	Carte 2D . . . . .	33
3.5.3	Résultats . . . . .	36
3.5.4	Dimension temporelle . . . . .	37
3.6	Scripts divers . . . . .	38
<b>III</b>	<b>Conclusion</b>	<b>40</b>
<b>1</b>	<b>Appréciation</b>	<b>40</b>
<b>2</b>	<b>Critiques et difficultés</b>	<b>41</b>
<b>3</b>	<b>Travail restant et évolution</b>	<b>41</b>
3.1	Objectifs en attente . . . . .	41
3.2	Points d'évolution . . . . .	41
<b>Références</b>		<b>42</b>
<b>Table des figures</b>		<b>44</b>
<b>Table des tableaux</b>		<b>44</b>
<b>Annexes</b>		<b>46</b>
<b>A</b>	<b>Annexe 1</b>	<b>46</b>
<b>B</b>	<b>Annexe 2</b>	<b>47</b>
<b>C</b>	<b>Annexe 3</b>	<b>48</b>

# Remerciements

Je tiens à remercier ici toutes les personnes m'ayant permis de réaliser ce stage et ayant contribué à la rédaction de ce rapport.

Tout d'abord je souhaite adresser mes remerciements à mon maître de stage Pascal VASSEUR pour m'avoir fait confiance dans la reprise de ce projet d'étude innovant établi sur 3 ans et de grande envergure puisqu' impliquant une multitude d'équipes pluridisciplinaires à travers la France. Notamment, un grand merci à Guillaume ALLIBERT et Cédric DEMONCEAUX appartenant respectivement aux équipes **SIS** du laboratoire I3S à Nice et **ERL ViBot** du laboratoire ImViA au Creusot, pour leur aide sur les aspects ressources et matériel ainsi que les séminaires qu'ils ont organisé et qui ont permis de croiser les connaissances de différents membres des équipes afin d'avancer plus efficacement dans nos tâches respectives.

Merci aussi tout particulièrement à Charles-Olivier ARTIZZU et Ahmed Rida SEKKAT, respectivement doctorants pour les laboratoire I3S et LITIS pour leur expertise et leur aide apportée au développement de solutions du projet lorsque nos intérêts se croisaient ; ainsi qu'à Fabio MORBIDI pour avoir organisé conjointement avec Cédric, Guillaume et Pascal les séminaires qui nous ont permis d'échanger nos connaissances.

Bien évidemment, merci à Dao ZHOU pour son travail fourni l'année précédente ainsi que la disponibilité dont il a fait preuve pour répondre à toutes mes interrogations et assister le transfert de connaissances et de ressources du projet.

Je remercie également l'équipe du laboratoire qui m'a accompagnée durant ce stage autant sur le plan technique qu'administratif ou organisationnel. Notamment Christophe POIX pour la gestion du matériel informatique, Juliette DUBOIS pour les demandes concernant le matériel et le bâtiment, Jean-François MAGNIER pour son soutien concernant l'utilisation à distance du réseau UPJV, mais également Mathieu BAUM au LITIS pour son accompagnement dans la partie administrative.

Enfin, je remercie mes collègues et amis stagiaires et/ou doctorants Daniel, Loic, Léo, Reda et Sarah sur qui je pouvais m'appuyer à tout moment et qui ont su rendre ce séjour Amiénois des plus agréables.

# Première partie

## Présentation

### 1 Organisme d'accueil

#### 1.1 Généralités

Localisé à Amiens, le MIS<sup>1</sup> est un laboratoire de recherche issu de la fusion en 2006 du CREA (Centre de Robotique, Électrotechnique, et Automatique) et du LaRIA (Laboratoire de Recherche en Informatique d'Amiens). Il opère dans les domaines de l'informatique, l'automatique, la robotique et la vision par ordinateur pour des thématiques de Sciences et techniques de l'information et de la communication (STIC). Les applications fréquentes que l'on retrouvera seront celles du véhicule, la cybersécurité, l'énergie, la musique, le patrimoine ou encore la santé.

Bien qu'affilié à l'UPJV<sup>2</sup>, le MIS est situé dans les locaux de l'ESIEE<sup>3</sup> Amiens et dispose de 2 étages dédiés dont l'accès est restreint au personnel autorisé car la zone est classées ZRR<sup>4</sup>. De plus, les chercheurs du laboratoire ont accès à un grand nombre d'instruments de mesure, de bancs techniques et de ressources informatiques de l'école.



(a) Vue aérienne de l'ESIEE



(b) Bâtiment du laboratoire MIS

FIGURE 2 – Ville d'Amiens

Sources : Courrier picard

Lorsque les besoins du laboratoire ne peuvent être satisfaits par les équipements disponibles sur le site de l'ESIEE, ceux-ci peuvent être assurés par l'une des antennes de l'UPJV dans la ville. L'école ESIEE ne partage cependant pas ses équipements et outils qui leur sont propres étant donné que le laboratoire est exclusivement affilié à l'UPJV.

1. Modélisation, Information & Systèmes

2. Université de Picardie Jules Verne

3. École Supérieure d'Ingénieurs en Électrotechnique et Électronique

4. Zone à Régime Restrictif

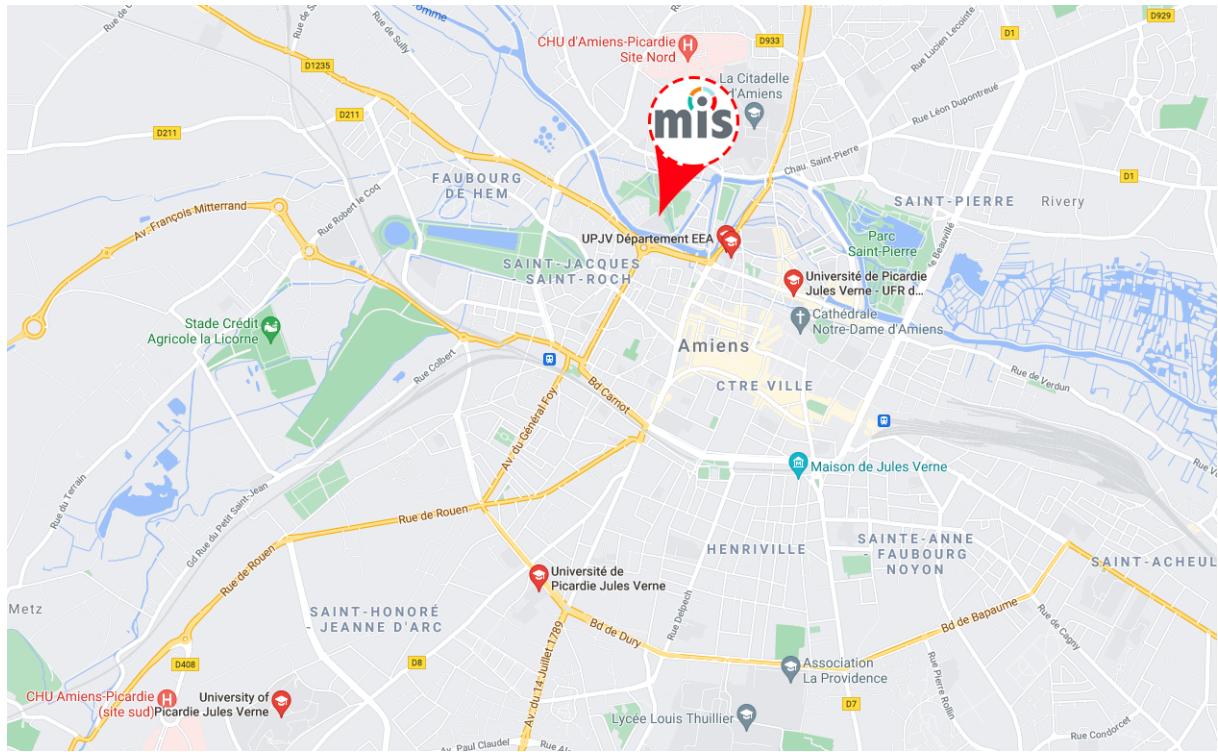


FIGURE 3 – Antennes de l'UPJV

Source : Google maps

## 1.2 Activité

Parmi les projets les plus imposants du MIS, on compte la reconstitution 3D de monuments historiques tels que la cathédrale d'Amiens : "e-cathédrale", qui a notamment permis au laboratoire d'assurer son identité pluridisciplinaire et sa visibilité à travers la couverture médiatique engendrée par un tel chantier. De plus, le laboratoire participe de manière croissante à l'innovation scientifique à échelle internationale en répondant à de plus en plus d'appels d'offres de projets collaboratifs européens. De manière analogue, le laboratoire soumet également de plus en plus de projets aux instances européennes dont certains ont été couronnés de succès, offrant aux équipes un fort apport en expérience et en visibilité.

## 1.3 Équipes

A la tête des 4 équipes de recherche *COVE* (Commande et Véhicules), *GOC* (Graphes, Optimisation et Contraintes), *PR* (Perception Robotique) et *SDMA* (Systèmes Distribués, Mots et Applications), le MIS s'appuie sur un effectif d'environ 80 personnes incluant 40 enseignants-chercheurs, 35 doctorants et 4 personnels administratifs et techniques, mais aussi sur un précieux réseau de partenaires industriels et académiques. Il n'est pas rare de voir des collaborations se former avec d'autres laboratoires européens dans le cadre de projets collaboratifs.

Il est également courant pour les projets nationaux d'impliquer des équipes de recherche réparties inégalement sur le territoire français ; comme cela est le cas du projet ANR<sup>5</sup> CLARA<sup>6</sup> dans lequel s'est inscrit mon stage. Dans ce projet j'ai été rattaché à l'équipe PR dirigée par mon maître de stage Pascal VASSEUR et ai été amené à travailler en collaboration avec une équipe du laboratoire I3S à Nice et une autre du laboratoire ImViA au Creusot (région Bourgogne). De plus j'ai également pu compter sur le soutien de Dao ZHOU, un ancien stagiaire du MIS désormais établi en Chine ayant travaillé auparavant sur mon projet, dont j'ai directement repris le travail.

5. ANR : Agence Nationale de Recherche

6. CLARA : Couplage Apprentissage et Vision pour Contrôle de Robots Aériens

## 1.4 Cadre de travail

Amiens, surnommée "petite Venise du Nord" pour ses nombreux canaux issus de la Somme, est une commune des Hauts-de-France et préfecture du département de la Somme. Capitale historique de l'ancienne Picardie et ville principale du département, Amiens témoigne par son style architectural et la préservation de son patrimoine, du passage de l'histoire sur plusieurs siècles et millénaires. Avec sa communauté d'agglomération, Amiens métropole regroupe plus de 400 000 habitants, la plaçant seconde ville la plus peuplée de sa région derrière Lille, et comprend de nombreux sites et monuments historiques tels que sa très connue cathédrale Notre-Dame, ses parcs archéologiques ou encore ses châteaux.



(a) Cathédrale et habitations



(b) Hortillonnages

FIGURE 5 – Ville d'Amiens

Sources : Wikipédia & Courrier picard

## 1.5 Contraintes sanitaires

De nombreuses mesures ont été prises et mises à jour pendant la durée de ce stage ce qui a fortement conditionné l'organisation du personnel et les règles d'utilisation du bâtiment et des équipements.

Tout d'abord, du gel hydroalcoolique a été mis à disposition dans la salle et il était demandé d'assurer une circulation de l'air aussi souvent que possible. L'effectif maximum dans les salles a été réduit ce qui a conduit la salle des stagiaires à n'accepter que 3 personnes en simultané. Cela a demandé une certaine organisation car durant ces 5 mois, de nombreux autres étudiants ont commencé puis fini leur propre période de stage. Cette organisation a été rendu plus facile grâce à un outil d'agenda en ligne nommé GRR (Gestion et Réservation des Ressources), accessible suite à la création de nos comptes UPJV par Christophe POIX, ingénieur en informatique.

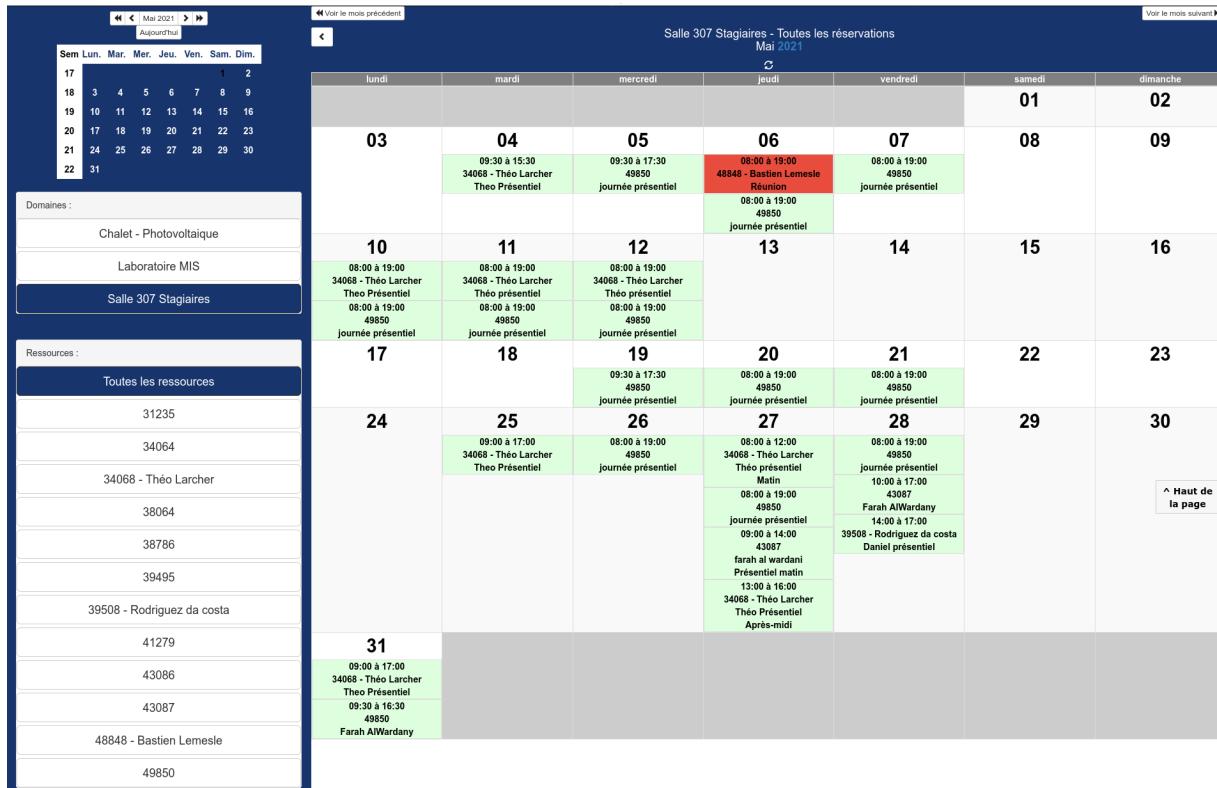


FIGURE 6 – Outil d’agenda en ligne GRR

## 2 Présentation du sujet de stage

### 2.1 Contexte du projet

Le projet est né en 2019 par la volonté de Pascal VASSEUR, Guillaume ALLIBERT et Cédric DEMONCEAUX de se mettre à collaborer sur un sujet d’étude commun et innovant. Ils déposèrent dans un premier temps un mini-projet auprès de l’Université Côte d’Azur (UCA) qui s’est ensuite transformé en le projet ANR CLARA. A partir de chacune de leurs spécialités (perception, vision omnidirectionnelle, commande), le trinôme décida de passer en revue l’état de l’art sur les drones autonomes puisqu’il s’agissait d’un projet innovant dans l’air du temps et son application à la navigation forestière est venue du constat qu’il s’agissait à l’époque (et aujourd’hui encore) d’un problème complexe non résolu.

Le projet a depuis grandi en taille et nombre de collaborateurs. Sur cette première moitié de 2021, je reprends les travaux effectués par Dao ZHOU, ancien stagiaire ayant travaillé sur le projet.

### 2.2 Sujet de stage

Le projet ANR CLARA commencé en 2019 est un projet collaboratif d’une durée de 4 ans (3 ans planifiés de base, étendus à 4 suite à la crise du COVID-19) impliquant les équipes d’Amiens, de Nice et du Creusot visant à développer un drone aérien autonome capable de naviguer en milieu forestier sans GPS en se basant sur un système de stéréovision omnidirectionnelle. Parmi les différentes fonctionnalités à développer, la création d’une carte pour la localisation et la navigation constitue un défi important. En effet, l’absence de structure géométrique régulière et ordonnée comme on pourrait en trouver en environnement urbain ou intérieur, implique d’utiliser des primitives différentes des approches classiques du SLAM. Il est envisagé ici d’utiliser l’implantation des arbres (leur position au sol) et leurs caractéristiques intrinsèques (diamètre, inclinaison, ...) pour estimer une carte 3D qui pourra être utilisée par la suite pour se repérer. Des travaux déjà réalisés au sein de l’équipe ont permis de réaliser une détection des arbres grâce à une approche par apprentissage et d’estimer leurs positions respectives.

# Deuxième partie

## Travail effectué

### 1 Cahier des charges

#### 1.1 Objectifs du projet

Dans ce stage, l'objectif consiste à améliorer ce module de détection des arbres dans une séquence d'images RGB omnidirectionnelles (éventuellement stéréo), de les caractériser (dimensions, positions, formes, ...) et de les intégrer à une carte 3D. Pour atteindre cet objectif, le travail se divisera selon les étapes suivantes :

- étude de l'état de l'art sur la navigation en milieu forestier et sur la cartographie visuelle
- amélioration du module de reconnaissance d'arbres et du sol en termes de précision et de robustesse
- développement d'un module de caractérisation 3D du sol et des arbres
- construction précise d'une carte 3D

L'objectif est d'allier les techniques liées à l'apprentissage et à la géométrie pour créer un outil rapide et robuste permettant l'obtention d'une carte précise et utile pour la localisation et la navigation. A minima, il est demandé de produire une carte de localisation 2D des arbres autour du drone en mouvement sans prendre en compte les aspérités du terrain.

#### 1.2 Cible du projet

Ce projet vise à être intégré dans une solution tout-en-un de drone autonome opérant dans un milieu forestier et s'adresse donc à des utilisateurs techniques du milieu de la recherche.

#### 1.3 Besoins et contraintes liés au projet

La partie R&D<sup>7</sup> est libre en termes de choix techniques et devra être achevée en priorité des productions annexes telles que la documentation ou les présentations. Les langages **Python 3** et **Bash** sont cependant pré-sentis comme langages de programmation principaux.

Étant donné que les travaux débuteront avec la reprise de ceux de Dao, le stagiaire a fait le choix de reproduire le plus possible à l'identique l'environnement de travail virtuel de Dao en ré-installant les mêmes paquets dans les mêmes versions (dans la limite du possible des compatibilités).

Il a été fait comme choix d'héberger le projet sur le GitHub du stagiaire avec la récupération des travaux de Dao sur son propre GitHub. Certaines ressources lourdes telles que les données d'entraînement et de test (collection d'images) ainsi que les environnements virtuels de forêts sont stockées en local sur la machine de travail.

Pour assurer une qualité de code satisfaisante, il a été décidé que des *linters*<sup>8</sup> et *testers*<sup>9</sup> pourraient être utilisés lorsque jugés pertinents.

#### 1.4 Périmètre du projet

Ce projet consiste à concevoir et développer un nouvel outil de reconnaissance et de localisation d'arbres sous la forme d'un POC<sup>10</sup> comprenant 2 briques majeures : un module de segmentation d'arbres et un module de localisation d'arbres. Le projet ne comprendra pas d'intégration à une plateforme de développement et ne nécessitera obligatoirement pas la réalisation de tests unitaires strictes. Cependant, toute méthode de développement visant à améliorer la qualité de réalisation est la bienvenue et le stagiaire pourra juger librement de celles qui conviennent le mieux au regard du facteur temps. Les parties prenantes sont le maître de stage et professeur des universités Pascal VASSEUR, les équipes de développement aux laboratoire I3S et ImViA, le laboratoire LITIS ainsi que de potentiels futurs développeurs reprenant les travaux du stagiaire.

---

7. Recherche et Développement

8. Outil d'analyse statique du code source

9. Outil de tests unitaires du code source

10. Proof Of Concept

## 1.5 Délais

Le projet débute le 01/03/2021 et se termine le 31/07/2021.

Ce projet étant fortement orienté recherche, certaines pistes pourront être privilégiées et d'autres abandonnées en fonction des résultats obtenus, en accord avec le maître de stage.

## 1.6 Méthodologie de gestion de projet

Des points d'avancée seront établis prioritairement lorsque le stagiaire en sentira le besoin, soit pour informer de l'avancée du projet soit pour alerter d'un problème ou d'un blocage. Si le besoin s'en fait sentir, le maître de stage pourra demander à faire des réunions supplémentaires.

A minima, un point hebdomadaire sera réalisé pour informer des avancées et difficultés rencontrées afin de pouvoir agir sur les choix techniques avec souplesse et pouvoir redéfinir les priorités du projet avec aisance, compte tenu de son aspect R&D.

De plus, plusieurs séminaires de l'équipe PR sont prévus dans l'année et seront accessibles en tant que spectateur. Au moins 1 participation sera demandée pour présenter les travaux du stagiaire durant ses 5 mois au reste de l'équipe et afin de tenir chaque membre au courant de ses activités et éventuellement croiser les connaissances et résultats de chacun.

# 2 Installation et reprise des travaux précédents

La première phase de ce stage a consisté en la récupération du matériel et des ressources nécessaires à la satisfaction des ambitions du projet. Cette étape ayant nécessité la collaboration de plusieurs parties dont certaines à l'extérieur du laboratoire, elle ne s'est pas rigoureusement inscrite en amont du travail de recherche et de développement mais s'est plutôt déroulée en parallèle du projet sur sa première moitié.

## 2.1 Matériel physique (*Hardware*)

La réalisation de ce projet a rendu nécessaire le fait d'avoir accès à une machine à fort potentiel calculatoire pour deux raisons.

La première est l'entraînement et le *fine-tuning*<sup>11</sup> de modèles *Deep Neural Network (DNN)*<sup>12</sup> récupérés dans les travaux de Dao ou parmi ceux trouvés sur le net lors de mes recherches bibliographiques.

La seconde est la manipulation du moteur de jeu [Unreal Engine 4](#) (UE4) ayant permis la création d'environnements forestiers virtuels suffisamment photo-réalistes pour pouvoir s'en servir afin de constituer des bases synthétiques<sup>13</sup> d'apprentissage et de test destinées aux modèles précédemment mentionnés.

Par conséquent il était important de posséder une carte graphique (*GPU*<sup>14</sup>) adaptée aux calculs intensifs ainsi que d'une quantité élevée de mémoire vive (*RAM*<sup>15</sup>) pour gérer les entraînements des algorithmes et la simulation en temps réel du moteur de jeu. Un processeur (*CPU*<sup>16</sup>) multi-coeurs était également important pour la parallélisation des processus. Après en avoir fait la demande, j'ai donc migré de mon poste d'origine vers une tour capable d'accueillir les composants adéquats suivants :

- CPU : Intel Xeon(R) E5-2609 v2 @ 2.50GHz (8 cores)
- GPU : Nvidia Quadro M6000 24GB
- RAM : 32Go Micron DDR3 1866MHz

---

11. Perfectionnement, affinage

12. Réseaux de Neurones Profonds

13. Base de données composée d'images d'environnements forestiers virtuels dites 'synthétiques' par opposition à des captures en environnement réel.

14. Graphics Processing Unit

15. Random Access Memory

16. Central Processing Unit



(a) GPU : Nvidia Quadro M6000 32GB



(b) CPU : Intel Xeon(R) E5-2609 v2

FIGURE 8 – Composants principaux

Sources : Amazon

Par ailleurs, une plateforme de calcul nommée *Matrix* était également à disposition des chercheurs et stagiaires du laboratoire pour des calculs extrêmement intensifs mais à laquelle je n'ai jamais eu besoin de recourir.

## 2.2 Matériel virtuel & logiciels (*Software*)

Durant mon stage je suis allé et venu entre deux distributions : **Ubuntu 20.04 LTS** et **Windows 10**.

La première, Ubuntu, m'a principalement servi pour le développement de solutions *deep learning*, pour la création de cartes, l'archivage de documents liés à la gestion de projet (*i.e.* Compte-rendu (CR) de réunions, journal de bord (JdB), notes etc...) et la gestion du GitHub. Je n'ai donc pas eu besoin de logiciel à proprement parler sur cette distribution si ce n'est l'éditeur de code Virtual Studio Code, cependant j'ai du installer un grand nombre de paquets Linux ou Python notamment pour satisfaire les prérequis des travaux de Dao.

La seconde, Windows, m'a surtout servi pour la manipulation des environnements virtuels sous l'éditeur d'UE4, la rédaction de scripts de commande pour interagir avec ces environnements, et la capture d'images pour mes bases de données virtuelles.

Pour servir cet équilibre, j'ai mis en place un *dual boot* sur mon poste avec une partition partagée par les deux systèmes d'exploitation (OS<sup>17</sup>) pour leur permettre de s'échanger des données entre eux, comme du code ou des images. De plus, puisque le télétravail a fait partie intégrante du mode de fonctionnement du laboratoire, j'ai configuré, avec l'aide du service Direction des Systèmes Informatiques (DSI), un accès à distance via *SSH*<sup>18</sup> (accès terminal) et *RDP*<sup>19</sup> (accès graphique) sur mon poste de travail pour chacun des OS, de manière à pouvoir travailler de chez moi à partir de ma machine personnelle. Le projet a nécessité d'utiliser des bases de données d'images capturées depuis l'UE4 Editor<sup>20</sup> (cf. 3.2.1) via la coordination du *plugin*<sup>21</sup> AirSim et de scripts Python et il était clair dès le début qu'il aurait été nécessaire de travailler sur mon poste à distance, faute de performances sur ma machine portable personnelle. Néanmoins, en cas d'urgence il m'était toujours possible d'accéder physiquement à la machine le temps de traiter l'incident.

Il est également à noter que la réception de la carte graphique ne s'est pas faite immédiatement au démarrage du stage puisqu'il a fallu en faire la demande d'acquisition. C'est lorsque celle-ci est arrivée qu'il m'a fallu migrer vers un autre poste car le mien était physiquement incapable d'accueillir le *GPU*. Par conséquent j'ai du refaire mes installations et configurer de nouveau les accès à distance ce qui a ralenti le démarrage du projet.

17. Operating System

18. Secure Shell

19. Remote Desktop Protocol

20. Logiciel d'édition de monde d'UE4

21. Extension, greffon



### Remarque

Afin d'être certain de pouvoir utiliser les travaux de Dao, desquels je suis parti, et qui se reposent eux-mêmes sur des productions tierces, j'ai été contraint d'utiliser d'anciennes versions de paquets. L'utilisation notamment de Tensorflow 1 a conduit à un manque de visibilité sur les métriques des modèles de *deep learning* car au vu du temps à disposition et de la complexité des projets tierces, il n'a pas été possible d'allouer un temps de formation suffisant à l'utilisation de la librairie pour pouvoir utiliser, ajouter et visualiser toutes les métriques que nous aurions voulu ajouter.

## 2.3 Travaux de Dao

En parallèle des installations logicielles et *hardware*, ce projet a démarré avec la récupération des travaux de Dao à partir de son GitHub dans lequel on retrouve plusieurs solutions liées à l'identification visuelle d'arbres dans des scènes forestières.

Ces solutions ont été consignées dans un rapport de stage duquel je suis parti pour faire un état des lieux des outils opérationnels déjà à disposition pour le projet, des points d'amélioration et des objectifs suivants à réaliser.

Durant son stage, trois aspects aidant à l'identification d'arbres ont été creusés : la détection par *bounding box*<sup>22</sup>, l'estimation de profondeur et la segmentation sémantique. Pour chaque, revenons rapidement sur les conclusions et les travaux légués.

- *Bounding box (bbox)*

L'identification par boîtes englobantes a été tentée avec le réseau Yolo v3 reposant sur un *backbone*<sup>23</sup> Darknet d'une part, et d'autre part avec le réseau AdapNet++, développé par Abhinav VALADA et al. [1] et reposant sur un *backbone* ResNet50 ; mais dans les deux cas la précision des détections était très moyenne, incluant des éléments du décor en arrière plan, en particulier pour les arbres dont les multiples branches sont visibles. De plus l'annotation pour un apprentissage supervisé est quant à elle très fastidieuse. Pour toutes ces raisons la méthode par *bbox* a été écartée.

- *Estimation de profondeur*

Afin de pouvoir localiser les arbres vis-à-vis du drone dans l'espace il était impératif de trouver un moyen de quantifier la distance les séparant. Dao aillant travaillé exclusivement à partir d'images, il a tenté une approche d'estimation de la profondeur en mode monoculaire via le réseau DenseDepth, développé par Ibraheem ALHASHIM et al. [2]. Le modèle initialement adapté aux scènes d'intérieur s'adapte parfaitement bien aux scènes synthétiques d'UE4 en montrant des estimations d'une précision impressionnante. Cependant, ça n'est pas le cas sur des images de scènes réelles issues d'internet du fait de l'introduction de bruit plus important et de conditions d'éclairage plus difficiles.



(a) Vue RGB



(b) Vue profondeur (vérité terrain)



(c) Vue profondeur (prédiction du réseau)

FIGURE 10 – Comparaison des vues de profondeur de la scène n°24 issue du monde virtuel UE4.16 de Dao

22. Boîtes englobantes

23. Architecture réseau squelette

- *Segmentation sémantique (SS)*

La segmentation sémantique a quant à elle été conduite grâce au réseau AdapNet++ et reposant sur un *backbone* ResNet50 ; où la différence s'est faite sur le choix du *dataset*<sup>24</sup>. Sur le *dataset* Freiburg, le réseau réussit à bien distinguer les éléments de nature différentes (arbres, chemin, ciel, herbe) mais sans forcément faire de distinction entre les arbres. Sur le *dataset* constitué par Dao, composé de plusieurs images de forêt récupérées d'internet, le réseau réussit à séparer les arbres espacés en particulier sur des images de bonne qualité à contraste suffisant.

Cependant pour pouvoir ré-entraîner le réseau en mode supervisé, il était nécessaire de disposer de *datasets* annotés, c'est-à-dire de posséder le double de chaque image où chaque objet qui nous intéresse est "colorié" d'une couleur unie de manière à ce que le réseau apprenne à les identifier individuellement. Cependant on se heurte ici au problème majeur de ces tâches d'apprentissage : les *datasets* annotés de forêts sous canopée sont très rares et il est souvent plus simple de les faire soi-même à partir d'environnements synthétiques. Pour cette raison, un premier environnement virtuel sous UE4.16 fut utilisé par Dao duquel il a ensuite été possible d'extraire des captures de scène pour constituer des bases d'apprentissage. Puis, grâce au *pluginUnrealCV*, il fut possible de modifier automatiquement les textures des différents éléments (arbres, sol, ciel...) du monde et même de faire une distinction entre les troncs d'arbres et les feuillages si besoin.

L'entraînement du réseau AdapNet++ sur cette base de données annotée a produit des résultats quelque peu approximatifs mais néanmoins intéressants.

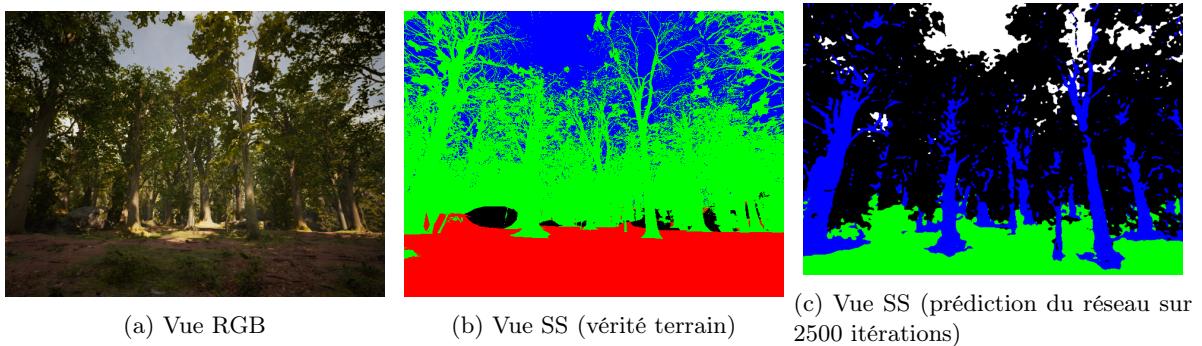


FIGURE 12 – Comparaison des vues segmentation sémantique de la scène n°24 issue du monde virtuel UE4.16 de Dao

Enfin, à partir de cette segmentation sémantique, Dao a développé une première méthode pour effectuer une segmentation par instance des troncs d'arbre grâce à des opérations morphologiques sur l'image et à des méthodes de *clustering*. Cependant cette phase du projet étant arrivée à la fin de son stage, la précision des *clusters* n'est pas toujours optimale et va même jusqu'à identifier plusieurs *clusters* sur un même tronc.

---

24. Base de données



FIGURE 13 – *Clustering* de troncs isolés de la scène n°24 du monde virtuel UE4.16 de Dao

Afin d'avoir à tout instant une vision globale sur le travail de Dao, j'ai également établi une *roadmap*<sup>25</sup>, pratique pour identifier d'un coup d'œil les pistes explorées et leur avancement (cf. Annexe 1).

## 2.4 Reprise des travaux

### 2.4.1 Algorithmes

Après la récupération des ressources de Dao depuis son dépôt GitHub, la première étape a été de se ré-approprier le projet en reprenant chaque brique développée (correspondant aux points précédemment abordés) pour tester leur bon fonctionnement et "nettoyer" le code en révisant les commentaires et en passant le code sous les *linters*. Cependant cette tâche s'est avérée moins triviale que prévue car il y avait peu de documentation sur la structuration du dépôt, et les commentaires de code étaient parfois en chinois. Afin de ne pas perdre trop de temps à déduire le fonctionnement complet du projet, voire à rester coincé, j'ai contacté Dao pour lui demander des précisions sur l'utilisation des différentes briques qu'il a développées. Nos échanges se sont étalés sur 1 mois car à cause du décalage horaire avec la Chine, du temps nécessaire pour systématiquement mettre en pratique les nouvelles indications et des disponibilités de chacun, nous étions limités à un rythme d'environ un échange tous les 2 jours.

A l'issue de cette période, le code était commenté et re-structuré et j'ai pu établir les performances pratiques des algorithmes. Nous avions notamment émis des doutes, Pascal et moi, sur la robustesse du réseau DenseDepth, censé estimer la profondeur d'images RGB, car les prédictions sur les images du monde de Dao étaient surprenamment bonnes. En reprenant les poids tels quels j'ai donc évalué le modèle sur des images réelles issues d'internet et comme mentionné dans la partie précédente, les prédictions ne sont pas aussi bonnes.

25. Feuille de route

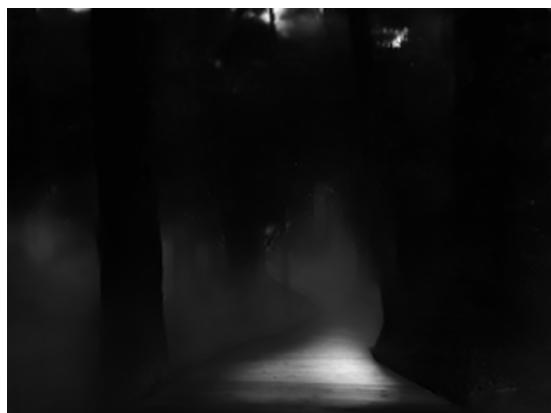
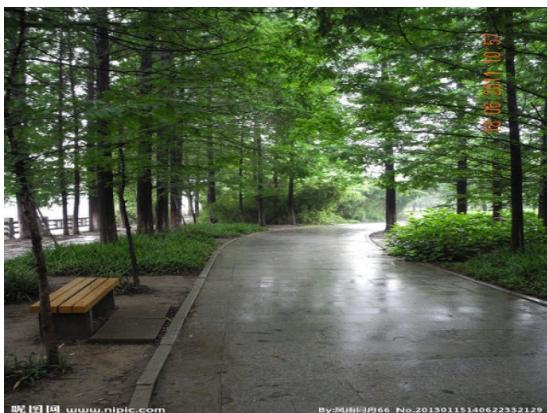


FIGURE 17 – Exemples de DenseDepth sur des images de scènes réelles (à gauche les images de référence ; à droite les prédictions du réseau)

On note que même si le réseau semble faire la distinction entre l'arrière plan et la projection des rayons du soleil sur le sol entre les arbres, il semble visiblement sensible aux variations de luminosité ce qui peut fortement induire en erreur ses prédictions sur l'estimation de distance d'une surface réfléchissante ou simplement plus éclairée que les zones qui l'entourent. De plus, la distinction entre les instances d'arbres devient d'autant plus floue que ces derniers sont rapprochés et nombreux et que leurs feuillages se superposent.

Le modèle AdapNet++ quant à lui, destiné à réaliser une segmentation sémantique des arbres d'une scène propose une précision intéressante sur des données synthétiques (cf. 2.3) à partir d'un entraînement suffisamment long (environ 2500 itérations) avec les paramètres suivants :

- *batch\_size* : 8
- *skip\_step* : 10
- *learning\_rate* : 0.001
- *power* : 0.9

Mais dès lors qu'on passe en entrée des images réelles, les prédictions se brouillent et il arrive de trouver plusieurs segmentations de classes sur un même objet comme sur l'exemple suivant où les classes "sol" et "arbre" sont confondus :

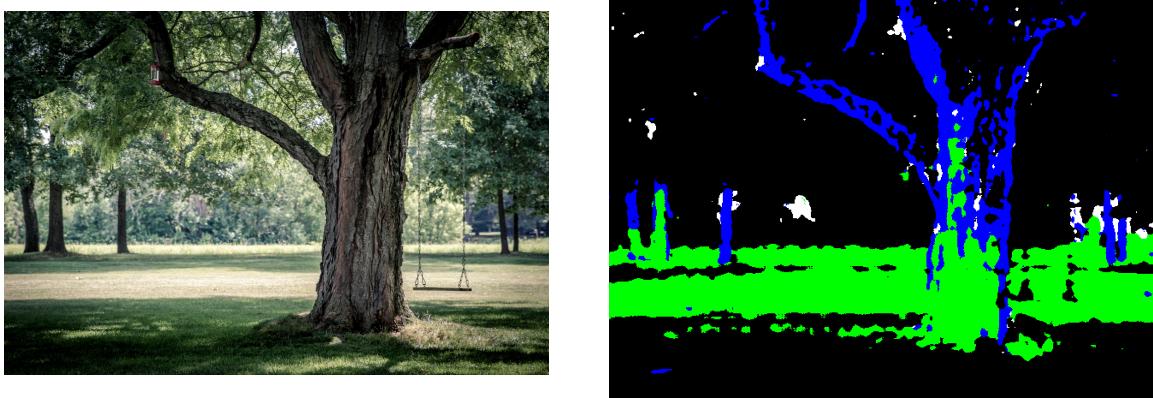


FIGURE 19 – Exemples de AdapNet++ sur des images de scènes réelles (à gauche les images de référence ; à droite les prédictions du réseau)

#### 2.4.2 Données et UnrealCV

En plus du code, j'ai également pu récupérer de petites bases de données ayant alimenté les réseaux AdapNet++ et DenseDepth (quelques centaines d'images tout au plus) ; mais aussi et surtout le monde UE4.16 contenant la forêt synthétique de Dao de laquelle sont tirées ces bases de données.

Couplé avec le *plugin* UnrealCV, il est possible dans cette simulation de se mouvoir librement dans l'espace, de placer ou supprimer des arbres ou tout autre objet (roches, végétation diverse, cours d'eau etc...), de changer la texture d'un objet et de prendre des captures d'écran programmables. Grâce à cela, il a été possible de constituer de nouvelles bases de données d'images synthétiques en vue perspective pour ré-entraîner les modèles.

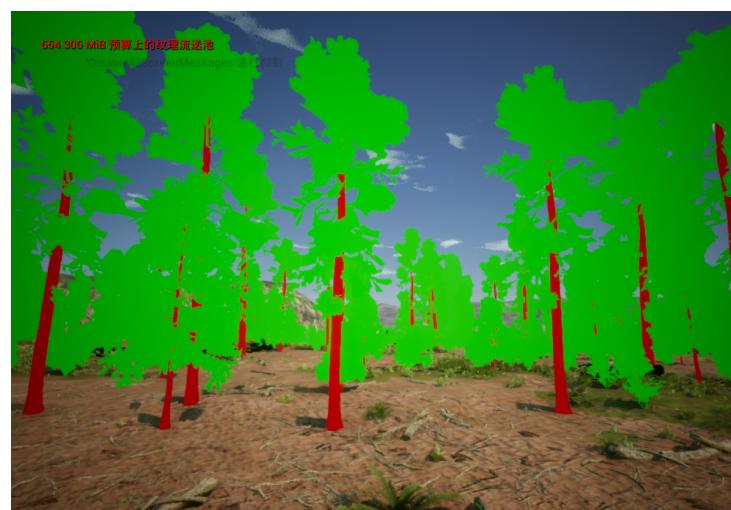


FIGURE 20 – Monde UE4.16 de Dao où les textures des arbres ont été modifiées pour rendre les troncs et feuillages d'une couleur unie

UnrealCV est un projet à destination des chercheurs en vision par ordinateur dans le but de les aider à construire des mondes virtuels sous Unreal Engine 4 et à les transformer facilement pour satisfaire les besoins de leurs projets. On trouvera parmi ces transformations, le changement de texture, l'application de filtres visuels, l'estimation de profondeur de scènes ou de encore de normales de surfaces. Son utilisation peut se faire de 2 manières différentes : soit intégrée à un projet UE4 compilé (auquel cas on peut interagir avec le monde directement par touches raccourcis ou par lignes de commandes externes) ; ou bien en tant que *plugin* de l'UE4 Editor (auquel cas son utilisation se fera exclusivement par lignes de commandes).

La prise en main de UnrealCV fut néanmoins compliquée car passé les soucis d'installation liés aux compatibilités de versions d'UE4, la solution logicielle ne comportait pas d'interface graphique et certaines commandes ne fonctionnaient pas car il s'agit d'un ancien projet expérimental dont le développement a été fortement réduit depuis 2020 ; réussir à identifier la source et la raison des erreurs de commandes a été un casse-tête en soi. J'ai tout de même fini par réussir à résoudre les *bugs* en tout genre et à faire fonctionner UnrealCV dans le monde de Dao dans lequel j'étais alors en mesure de faire toutes les actions attendues.

## 2.5 *Datasets* et environnements UE4

Comme dans tout projet comportant de l'apprentissage machine, se pose en priorité la question des données d'apprentissage. Nous avons constaté avec Pascal VASSEUR que tout comme lors du stage de Dao, les *datasets* de forêts annotées pour de la segmentation sémantique se font rares lorsqu'il s'agit de prises de vues sous canopée. Une des raisons majeures est que l'annotation manuelle de telles données représenterait une tâche gargantuesque, contrairement à des vues aériennes où la dimensionnalité peut grossièrement être réduite de 3 à 2.

Quelques sources telles que le [SFU Mountain Dataset](#) existent cependant, mais le contexte de l'environnement n'est pas exactement adapté à notre problématique puisqu'il s'agit de clichés pris par un robot terrestre le long d'un sentier. Or notre objectif est de réussir à assurer une navigation aérienne au cœur d'un environnement forestier sauvage. D'autres projets tels que [3] ou [4] proposent des méthodes de navigation de drone entre les arbres mais expliquent avoir constitué leur propre base de donnée en allant eux-mêmes effectuer les acquisitions sur le terrain.

### 2.5.1 Choix des données

Nous avions donc deux choix à notre disposition :

1. Créer notre propre *dataset* en allant effectuer des acquisitions dans la forêt la plus proche
2. Contacter les auteurs des rares papiers faisant mention d'un *dataset* 'maison' sans garantie de réponse et encore moins d'obtention
3. Utiliser des données synthétiques issues du moteur de jeu UE4

La solution n°2 est toujours viable car il est tout à fait possible de s'appuyer sur la coopération de la communauté scientifique mais nous avions des connaissances a priori qui ont orienté notre choix différemment.

Tout d'abord, il était prévu que l'équipe de Nice effectue ces acquisitions dans une forêt du Sud-Est de la France à l'aide de deux caméras omnidirectionnelles montées sur un même système. Cela aurait permis de bénéficier de données réelles stéréoscopiques avec la complexité propre aux captures réelles et ainsi de pouvoir entraîner directement nos modèles d'estimation de profondeur et de segmentation sémantique sur un jeu de données adapté. Malheureusement, il s'est avéré que du à des contraintes de temps et de disponibilité des membres de l'équipe, cette phase d'acquisition n'a finalement pas pu être faite sur la période de mon stage. La solution n°1 était dans tous les cas mise de côté.

Cependant, Guillaume ALLIBERT et Charles-Olivier ARTIZZU respectivement enseignant-chercheur et doctorant de l'équipe de Nice, possédaient également un environnement virtuel forestier sous UE4.26, récupéré du marché de la communauté Unreal grâce à son entreprise créatrice [MAWI United GmbH](#) et intitulé "MW Mountain Redwood Trees Forest Biome", ou plus simplement "Redwood Forest". L'équipe se servait alors déjà de cette carte comme base de travail pour le développement de la partie navigation du drone autonome forestier, et a gentiment accepté de nous y donner accès par le biais d'un compte Unreal Engine partagé.

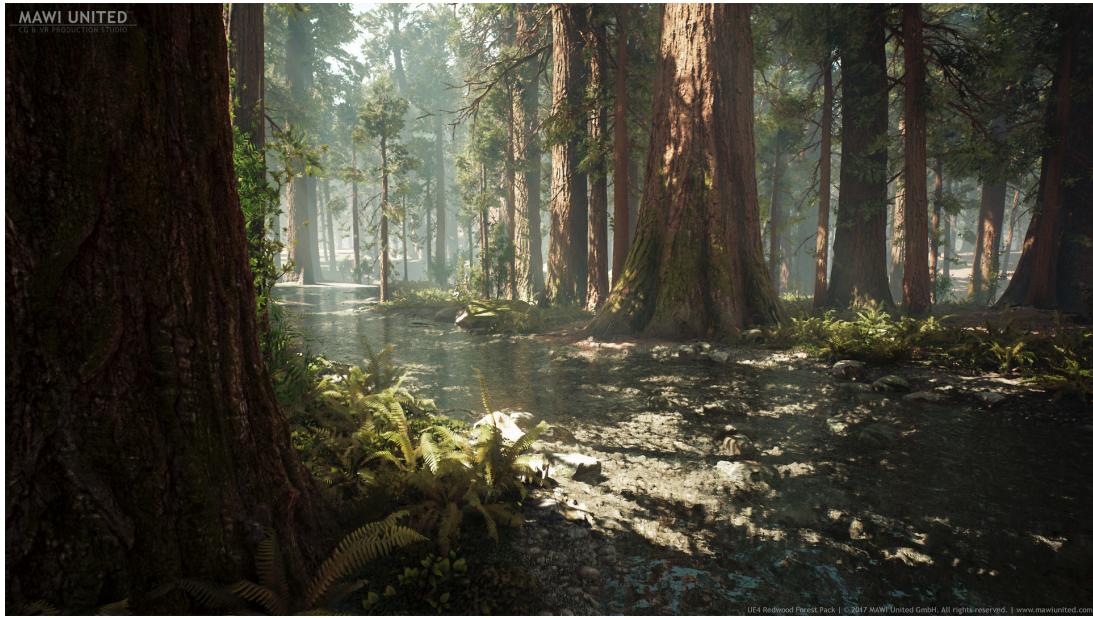


FIGURE 21 – Monde UE4.26 Redwood Forest

Ajouté à l'environnement UE4.16 de Dao, nous avions alors 2 mondes photoréalistes à notre disposition pour entraîner nos modèles, et étant donné que le cœur du sujet consistait en la création de carte, et qu'il restait encore de nombreux développements additionnels à la détection d'arbres, notre choix s'est porté sur la solution 2 : entraîner nos modèles sur des données synthétiques, comme l'avait fait Dao avant moi.

### 2.5.2 Comparaison des mondes UE4

Bien que nous ayions eu à notre disposition deux mondes virtuels de forêts photoréalistes, nous nous sommes tournés vers "Redwood Forest" plutôt que vers celui de Dao pour des raisons cruciales.

- **Version du monde**

Il faut comprendre que les produits d'Unreal sont tous liés par leurs numéros de version. Chaque monde ou *plugin* développé par un tiers ou Unreal eux-mêmes sera garanti de fonctionner pour des versions spécifiques d'UE4 car les mises à jour du moteur de jeu ne sont pas toujours rétro-compatibles. Au moment de mon stage, la version la plus récente du moteur en version 4 était la 4.27, ce qui signifiait que la communauté de développeurs Unreal était particulièrement active sur les versions 20+. Un désavantage notable pour le monde de Dao qui lui est figé en version 4.16.

Il est théoriquement possible de mettre à niveau tout projet (monde, extension... etc) soit par conversion automatique de l'UE4 Editor, soit en modifiant et en re-compilant manuellement les parties impactées. Mais parfois, certains *assets*<sup>26</sup> ne sont tout simplement pas compatibles avec les nouvelles versions du moteur de jeu ce qui conduit à un échec de la mise à niveau automatique et demanderait grossièrement de tout reprendre à la main dans la version voulue. Malheureusement, ce fut le cas avec le monde de Dao qu'il était impossible de passer en version ultérieure, du moins avec mes compétences techniques limitées car nouvelles sur le sujet.

- **Compilateur**

Ajouter à cela le fait que les projets Unreal reposent essentiellement sur du langage C++ qui demande à être compilé avant de pouvoir constater les changements dans l'UE4 Editor. Or, lorsqu'il s'agit de modifier manuellement certains fichiers de code, il est par conséquent nécessaire de re-compiler le projet à la main par l'intermédiaire de **Microsoft Visual Studio** (MSV)<sup>27</sup> ; mais là encore la version de MSV importe au regard de la version d'Unreal dans laquelle est développé le projet à compiler. Autrement dit, chaque version d'Unreal supportera ou non certaines versions du compilateur C++ de MSV.

- **Plugins pour la vision par ordinateur**

26. Terme désignant des ressources graphiques de moteurs de rendu 3D en général

27. MSV est un ensemble complet d'outils de développement (IDE) permettant de générer des applications web, des services web, des applications bureautiques ou mobiles etc...

De plus, de manière analogue à Dao qui utilisait UnrealCV, l'équipe de Nice utilise le *plugin AirSim*, développé par Microsoft sur la base d'UnrealCV, pour simuler leur drone dans le monde "Redwood Forest". Ce plugin contient également une bibliothèque d'outils utiles à la vision par ordinateur (puisque dérivé d'UnrealCV) et cela me sera utile plus tard. Il n'est cependant pas compatible avec les versions 4.16 et antérieures d'Unreal et est donc condamné à être utilisé sur le monde "Redwood Forest".

- **Projection des captures**

Enfin, les acquisitions via UnrealCV sur le monde de Dao produisent des captures en vue perspective ; c'est-à-dire qu'il n'y a pas de notion d'omnidirectionnalité de scène puisque cette dernière est observée comme à travers une fenêtre rectangulaire. Or nous nous intéressons dans ce sujet d'étude à des vues capables de représenter une scène dans ses 360 degrés d'angle telles que la vue **equirectangulaire**.

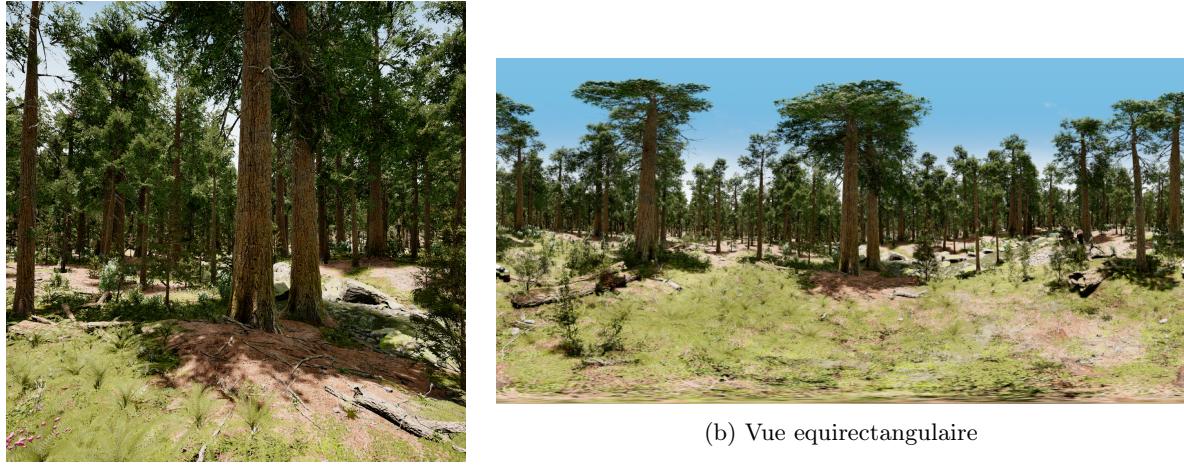


FIGURE 23 – Exemple comparatif des deux types de vue. Ici, la vue perspective correspond au centre de la vue equirectangulaire.

Dans l'UE Editor il est possible d'activer un *plugin* natif appelé "Panoramic Capture Tool" [5] effectuera alors des captures en vue equirectangulaire de la scène. Cependant le temps de capture est démesurément long (de 1 à 2 minutes sur ma machine de travail) ce qui ne constitue pas une solution confortable pour générer à la volée nos bases de données. Nous savions à l'avance que nous aurions certainement besoin d'en générer plusieurs en fonction des changements apportés à l'environnement virtuel, cette solution n'était donc pas une option.

Par ailleurs UE4.16, UnrealCV n'est pas doté de capacité de capture equirectangulaire ce qui aurait causé problème pour capturer des vues equirectangulaires du monde de Dao. Mais sous UE4.26, Charles-Olivier a été en mesure d'obtenir des projections grâce à la flexibilité de AirSim et à l'aide de **Blender**<sup>28</sup> ; la technique étant d'effectuer plusieurs captures d'une scène donnée via une seule caméra virtuelle dont on change l'orientation et de reconstruire la projection equirectangulaire via Blender.

Pour toutes ces raisons, nous avons fait le choix de façonnez notre développement autour du monde "Redwood Forest" et de AirSim pour toute la durée du stage.

## 2.6 Ralentissements au démarrage du projet

Il est important de comprendre que les sections abordées jusqu'ici se sont déroulées sur de longues périodes de temps car de multiples désagréments sont venus se greffer de manière inopinée au projet, en ralentissant la progression.

28. Logiciel de modélisation 3D

### • Unreal Engine 4

L'un des plus gros ralentissements est dû à l'installation et l'utilisation d'UE4 sur les postes de travail. Le logiciel est disponible à la fois sur Linux et Windows, suivant des modalités d'installation et d'utilisation différentes mais il s'est avéré qu'il a fallu abandonner la première installation Linux à cause de l'impossibilité de récupérer la carte "Redwood Forest". En effet, les mondes créés par des internautes et mis à disposition de la communauté sont parfois accessibles via des archives ouvertes accessibles en téléchargement grâce aux créateurs ; mais la plupart du temps elles doivent être téléchargées à travers l'**Epic Games Launcher**, le lanceur de la société Epic Games, propriétaire d'Unreal Engine. Or ce *launcher* n'est supporté que sous Windows ce qui signifie que notre seul moyen de récupérer la carte "Redwood Forest" acquise par Guillaume et son équipe, est d'utiliser UE4.26 sur une partition Windows 10.

Au moment de cette découverte, l'installation d'UE4 sous Linux avait déjà été faite car c'est une des premières actions que j'ai entreprises et que je n'étais pas au courant de l'alternative de Guillaume.

De plus, le launcher souffrait d'un bug de *RAM overflow*<sup>29</sup> lorsque je le lançais sur mon poste de travail contenant le GPU alors qu'il était relié à internet. Ce problème a persisté pendant 1 mois sans que nous comprenions, Pascal et moi, d'où cela pouvait venir jusqu'à ce qu'avec l'aide apportée par Jean-François MAGNIER du service réseau de l'UPJV, on arrive à déterminer qu'il s'agissait d'un problème de configuration de proxy au niveau de l'OS. Ce déblocage fut essentiel car il permit de gagner énormément en fluidité de travail puisqu'il était à présent possible d'utiliser le *launcher*, et donc de lancer le monde "Redwood Forest" tout en utilisant internet pour se former à l'utilisation de l'UE4 Editor et faire des essais de manipulation.

Puis est venu le temps d'apprentissage nécessaire à l'utilisation de l'UE4 Editor. J'avais l'avantage de connaître le fonctionnement général des logiciels de modélisation/animation 3D de par certains projets personnels ; mais il était clair que pour réussir à faire mes captures de scène et modifier mon environnement à souhait, j'allais avoir besoin d'une formation. C'est pourquoi dès Avril j'ai pris contact avec Charles-Olivier directement via LinkedIn pour qu'il puisse m'expliquer le fonctionnement du monde, la signification des objets de rendu, le paramétrage des lumières et tout autre aspect relatif à l'utilisation de l'UE4 Editor et du *plugin* AirSim. Il n'était alors pas rare que nous programmions des réunions en visio-conférence via WhereBy pour s'entre-aider.

Estimation du temps cumulé demandé : de l'ordre du mois.

### • Paquets Python et versionnage

Une seconde difficulté majeure a été de réussir à concilier les versions de tous les paquets Python utilisés dans le projet ; à la fois ceux issus des travaux de Dao (dont j'avais, en amont, demandé la liste détaillée) et les miens. Cependant il n'a pas été possible de reproduire à l'identique les environnements de travail virtuel au regard des nouvelles versions de Python et des paquets annexes tels que *jupyter*.

Trouver des versions de dépendances à des paquets principaux comme *tensorflow* 1.14.0 a été particulièrement compliqué et j'ai du me résoudre à utiliser conjointement 2 gestionnaires de paquets pour mon environnement virtuel : **Anaconda** pour la création de l'environnement et la gestion principale ; et **virtualenv** pour l'ajout rapide et quotidienne de paquets communs.

Estimation du temps cumulé demandé : de l'ordre de la semaine.

### • GPU Nvidia Quadro et Tensorflow 1

Un autre point à mentionner a été l'installation des paquets Linux nécessaires au bon fonctionnement et à l'utilisation effective de la carte graphique Nvidia Quadro M6000 32GB par Tensorflow 1 notamment pour les sessions d'apprentissage des modèles de *deep learning*. Lorsqu'ils sont insérés après l'installation de l'OS, les GPU Nvidia nécessitent que les utilisateurs Linux gèrent eux-même la récupération et l'installation des drivers standards (*nvidia-driver-450*) et spécifiques (technologies CUDA pour le deep learning par exemple, abrégé "cuDNN"). La manipulation est toutefois bien guidée par la documentation Nvidia qui possède une communauté Linux très active.

Les choses se sont cependant compliquées pour l'utilisation du GPU par les modèles de *deep learning* car ceux-ci reposent sur une architecture Tensorflow 1.14 et cette version, populaire en son temps, devient plus problématique de nos jours pour réussir à trouver les bonnes versions à la fois des paquets Linux relatifs à nos projets mais aussi des drivers compatibles. En particulier, il faut veiller à définir la variable

---

29. Dépassement de mémoire (vive)

d'état `os.environ['CUDA_VISIBLE_DEVICES']` à une valeur correspondant au numéro d'ID de notre carte graphique. Dans mon cas, j'ai du définir cette variable sur 0 puisque ma machine fonctionnait en mode *single-GPU*<sup>30</sup>.

Estimation du temps cumulé demandé : de l'ordre du jour.

- **Corruption de partition**

Enfin, un évènement majeur et potentiellement grave est venu paralyser le développement courant Avril. Suite à une opération de redimensionnement des espaces de partitions pour ajuster celle partagée entre mes deux OS, la taille des espaces mémoires alloués à mes partitions Linux n'étaient soudainement plus bien reconnues, résultant en une impossibilité de démarrage sur mon Ubuntu et même de lecture de mes données depuis une clef *bootable*<sup>31</sup>. Fort heureusement, j'ai été en mesure après de nombreuses manipulations de récupération de récupérer mes données et restaurer mes partitions qui n'avaient au final pas subi d'altération. L'opération aurait cependant pu coûter de nombreuses heures de travail.

Estimation du temps cumulé demandé : de l'ordre du jour.

## 3 Développement

### 3.1 Synchronisation des caméras

La configuration physique initialement imaginée par Pascal et l'équipe de Nice était d'utiliser deux [caméras 360 Ricoh Theta](#) montée sur un même support pour effectuer des captures simultanées. Mais pour qu'un tel système fonctionne, la notion de simultanéité doit être parfaitement assurée afin de minimiser au maximum l'écart temporel entre les captures des caméras gauche et droite. D'apparence facile, la réponse à cette problématique est en réalité plus complexe qu'il n'y paraît au premier abord à cause du degré de précision qu'il faut réussir à atteindre.

Bien que cette partie du projet fut au final entièrement reportée sur l'équipe de Nice responsable de l'acquisition, j'ai tout de même commencé par effectuer des recherches sur des méthodes de synchronisation afin de saisir l'enjeu des contraintes à bas niveau du projet et proposer des pistes de travail aux autres équipes.

#### 3.1.1 Astuces

Plusieurs méthodes et astuces existent pour réduire le temps de latence à bas coût. Les astuces les plus basiques consistent à garantir une mise sous tension simultanée des caméras ou un lancement mécanique simultané de l'enregistrement. Le problème est que cela ne garantit une synchronisation des captures que dans un monde parfait où les caméras se comportent strictement à l'identique ce qui n'est généralement pas le cas. Des décalages au niveau *software* peuvent intervenir, surtout au démarrage, ce qui induirait tout de même un décalage entre les trames.

Cette méthode est cependant plus susceptible de fonctionner sur un montage *hardware* personnalisé où l'on a un contrôle total sur la mise sous tension de chaque composant électronique. Cette configuration "DIY"<sup>32</sup> est généralement celle préférée par les possesseurs de matériel Arduino souhaitant fabriquer un montage stéréo à bas coût. Pour un montant inférieur à 200€ ce type de montage est possible, cependant dans le cadre de notre projet nous possédions déjà les caméras et il n'était pas prévu d'en changer.

#### 3.1.2 Generator-locked

Pour une réduction plus importante et sûre du décalage entre 2 captures, la plupart des caméras haut de gamme et professionnelles sont équipées d'une technologie appelée *Gen Locking* qui consiste à utiliser un signal émis par un générateur de synchronisation, une caméra ou une régie pour la synchronisation d'appareil d'enregistrement. Dans notre cas, cela signifierait utiliser le flux vidéo d'une de nos caméras en mode stéréo pour calibrer la fréquence de capture de l'autre. De telles caméras synchronisées seraient alors dites "*genlocked*" (à quelques nanosecondes près). Malheureusement, cette technologie n'est pas disponible sur toutes les caméras et les Ricoh Theta S n'en font pas partie.

---

30. GPU unique

31. Clef USB possédant une version allégée d'un système (Linux) et agissant comme une disque dur sur lequel on peut démarrer

32. "Do It Yourself", désigne une méthode artisanale de faire quelque chose

### 3.1.3 Post-traitement

Enfin il est toujours possible d'effectuer du post-traitement sur les flux vidéos afin de recaler les flux vidéos entre eux en comparant les temps de réception de chaque trame. Prarthana SHRESTH et al. proposent notamment dans [6] de se baser sur l'estimation d'un décalage initial de lancement de capture entre deux caméras, et de compter sur le fait qu'elles effectuent des captures à la même fréquence, pour estimer à bas coût le recalage temporel nécessaire pour synchroniser les flux audio et vidéo.

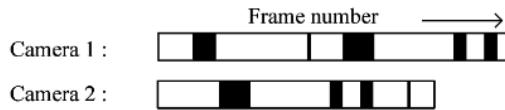


Fig. 2. Before synchronization: the recordings from two cameras with respect to the frame number. Some of the extracted features are represented in black.

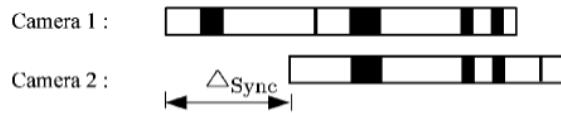


Fig. 3. After synchronization: the extracted features are matched to compute the synchronization offset, denoted by  $\Delta_{\text{Sync}}$ .

FIGURE 24 – Schéma extrait de [6] démontrant l'intérêt de baser le recalage des flux vidéos sur un décalage initial.

A cela on peut imaginer des contrôles de décalage périodiques des flux vidéo et audio pour s'assurer qu'il n'y ait pas de phénomène de décalage de la fréquence d'acquisition au cours du temps. Cela pourrait se traduire par une nouvelle comparaison des trames de capture, ou bien encore par l'estimation d'une distance entre 2 images ou 2 signaux audio capturés proches dans le temps pour déterminer s'il s'agit bel et bien de trames identiques (cette deuxième option pourrait cependant se révéler coûteuse en temps de calculs).

## 3.2 Simulateurs et captures

La tâche de la synchronisation des captures caméra étant reportée sur l'équipe de Nice, il était temps de nous atteler à la constitution de nos *dataset* synthétiques à partir du monde "Redwood Forest" sous Unreal Engine 4.26.

### 3.2.1 Unreal Engine 4 Editor

L'UE4 Editor est l'éditeur de niveaux de la plateforme Unreal Engine 4 et cela a été l'interface principale pour interagir avec le monde virtuel contenant notre forêt et se présente de la manière suivante.

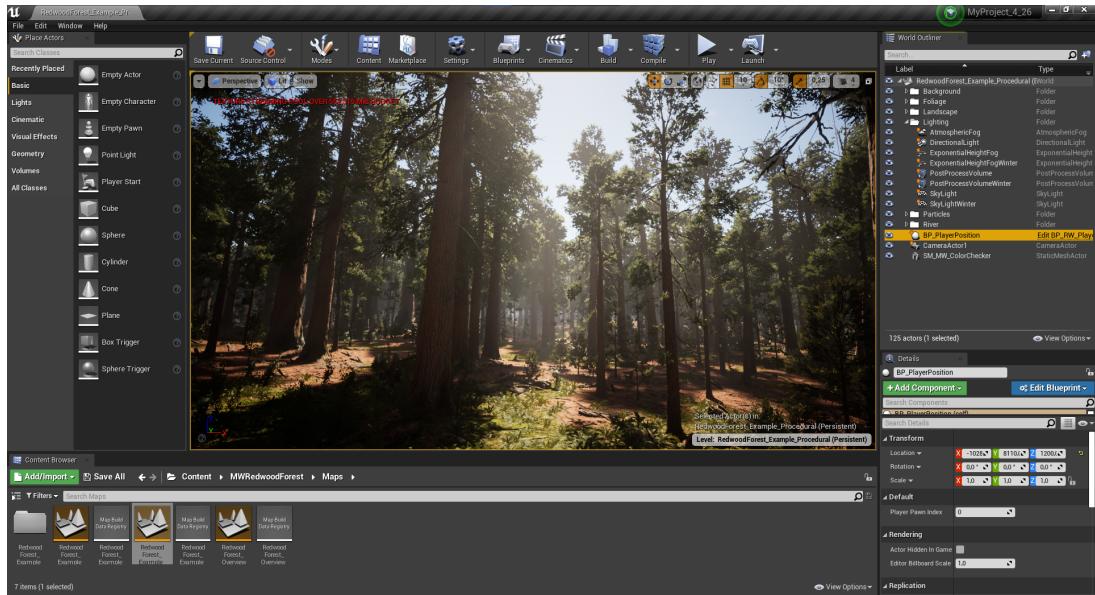


FIGURE 25 – Editeur de monde d'Unreal Engine 4.26 (exemple sur "Redwood Forest")

L'architecture des logiciels d'édition 3D ou d'édition de jeux est assez similaire dans le monde et fort heureusement pour moi qui ai un peu d'expérience avec ces logiciels, Unreal n'y fait pas exception et s'organise comme suit.

Au centre de la fenêtre se trouve le rendu de notre monde en temps réel. Il est interactif et l'on peut s'y déplacer comme bon nous semble pour observer, placer des objets, effectuer des mesures etc... Le rendu représente ce que l'utilisateur final verra lorsqu'il utilisera le produit fini, à la différence que le temps ne s'y écoule pas puisque l'objectif est de donner un aperçu 'figé' de son environnement.

Au dessus du rendu, on retrouve la barre d'outils permettant d'effectuer des actions en rapport avec le dit rendu ou le projet en entier. Lancer la simulation, sauvegarder le projet, changer les angles de vue du rendu, paramétrier les caméras... toutes ces actions passent par cette barre d'outils.

Sur la partie gauche on retrouve les différents objets de base pré-enregistrés par Unreal et qu'il est possible d'ajouter au monde. Organisés par catégories, ils constituent une véritable bibliothèque d'outils communs qu'on est certain de vouloir utiliser à un moment donné dans notre projet. Son objectif est de faire gagner du temps aux développeurs en fournissant des squelettes pré-fabriqués.

A droite, on trouvera sur la moitié supérieure un inventaire des objets présents dans le monde. L'organisation de ces éléments ressemble à une architecture par dossier et est définie par le développeur. Chaque objet est sélectionnable ce qui donne accès dans la partie inférieure à ses caractéristiques (nom, tailles, position, couleur, texture, dynamique... etc)

Enfin dans la partie inférieure de la fenêtre, on retrouvera la vraie architecture par dossiers de notre projet, contenant nos fichiers d'*assets*, de monde et ressources en tout genre. Si le projet est identifié comme une projet "C++" on trouvera également les fichiers de code. Par ailleurs, cet espace peut être partagé avec une console de commande destinée à taper des instructions dans le langage de l'UE4 Editor.

### 3.2.2 AirSim

AirSim est un simulateur *open-source* et multi-plateforme de drones et de voitures construit sur la base d'Unreal Engine (et maintenant Unity également) et développé par les équipes de Microsoft à partir

du projet UnrealCV pour les versions d'UE4 4.17 et supérieures. Son implémentation se traduit sous la forme d'une extension à un projet UE4 destiné à aider les chercheurs à développer des algorithmes d'apprentissage par renforcement pour la conduite autonome, à conduire des expériences en vision par ordinateur, en *deep learning* et bien plus encore.

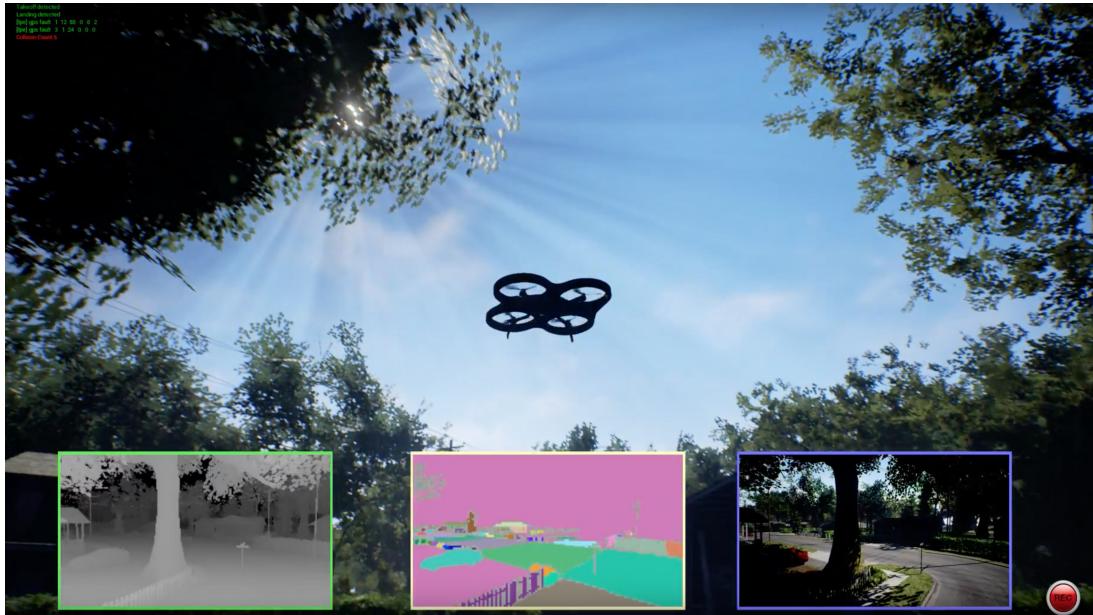


FIGURE 26 – Simulation de drone sous AirSim.

Source : [HelicoMicro.com](http://HelicoMicro.com)

La facilité de prise en main est une des composantes clefs qui fait la réussite de ce simulateur puisqu'il bénéficie d'une documentation détaillée [7], de vidéos tutorielles et offre la possibilité d'être manipuler au choix par une API<sup>33</sup> C++ ou Python, en fonction de l'application visée. La communauté autour de AirSim est aujourd'hui très vivante, en témoignent les nombreux projets indépendants que l'on trouve sur internet y faisant référence ou encore l'activité constante du GitHub du projet, ce qui augmente le dialogue autour de cette extension.

### 3.2.3 Création des données

#### • Choix de la méthode

Notre objectif était de construire une carte de localisation des arbres et nous avions des réseaux de neurones entraînés à reconnaître spécifiquement des arbres. Pour effectuer de l'apprentissage supervisé il allait nous falloir construire les données labellisées et pour cela deux choix s'offraient à nous :

1. Se servir du *plugin* AirSim pour extraire des vues sémantiquement segmentées
2. Se servir de l'UE4 Editor pour modifier le monde "Redwood Forest"

La première option est rendue possible par le fait que AirSim embarque avec son simulateur des solutions de vision de par ordinateur permettant de fournir à la fois des vues RGB, des cartes de profondeur et des segmentations sémantiques de scènes. Cependant, la segmentation sémantique proposée ne fait pas de distinction entre les types de végétaux qui sont, in fine, tous rangés dans la catégorie "végétation". Or nous aurions des difficultés à exploiter une telle segmentation étant donné la densité de la végétation dans notre monde "Redwood Forest". Nous aurions fait face à des problèmes de chevauchement de classes, sans réussir à distinguer les objets individuellement ni quelle relation de plan ces objets auraient eu entre eux (premier plan ? arrière-plan ? entre les deux ?).

La seconde option se traduisait par la modification des textures des arbres et autres objets ainsi que la simplification de l'éclairage de la scène de manière à obtenir un "monde label" duquel il n'aurait plus suffit que de faire des captures, de manière classique. De plus, la flexibilité de cette solution permettait de "peindre" les arbres de la couleur désirée, ouvrant potentiellement à la possibilité de constituer des

33. Application Programming Interface : interface de programmation d'applications

*datasets* pour la segmentation par instances directement et ainsi se passer d'une étape de *clustering* comme l'avait fait Dao pour identifier les arbres individuellement.

Après consulté Pascal puis Charles-Olivier qui avait pris l'habitude de manipuler le monde virtuel nous avons décidé de nous orienter vers l'option n°2 qui nous a parut plus avantageuse.

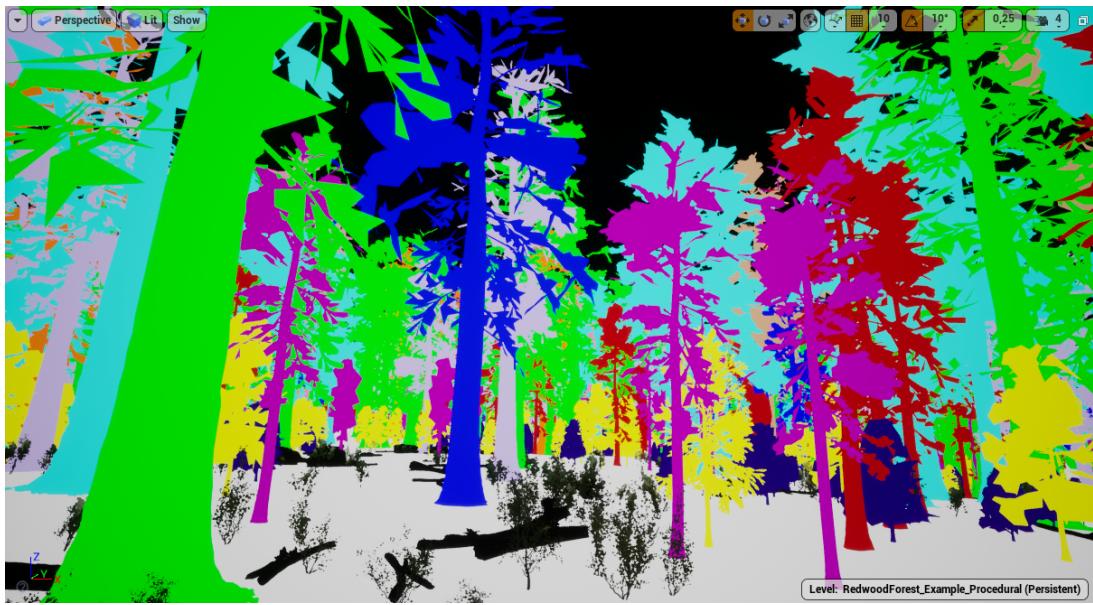


FIGURE 27 – Monde label issu du monde "Redwood Forest".

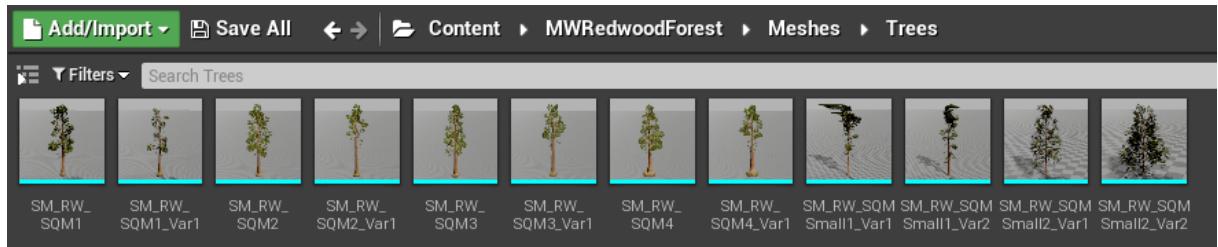
#### • Mise en pratique

Pour comprendre la mise en pratique de cette solution il faut d'abord comprendre la structuration de l'environnement virtuel.

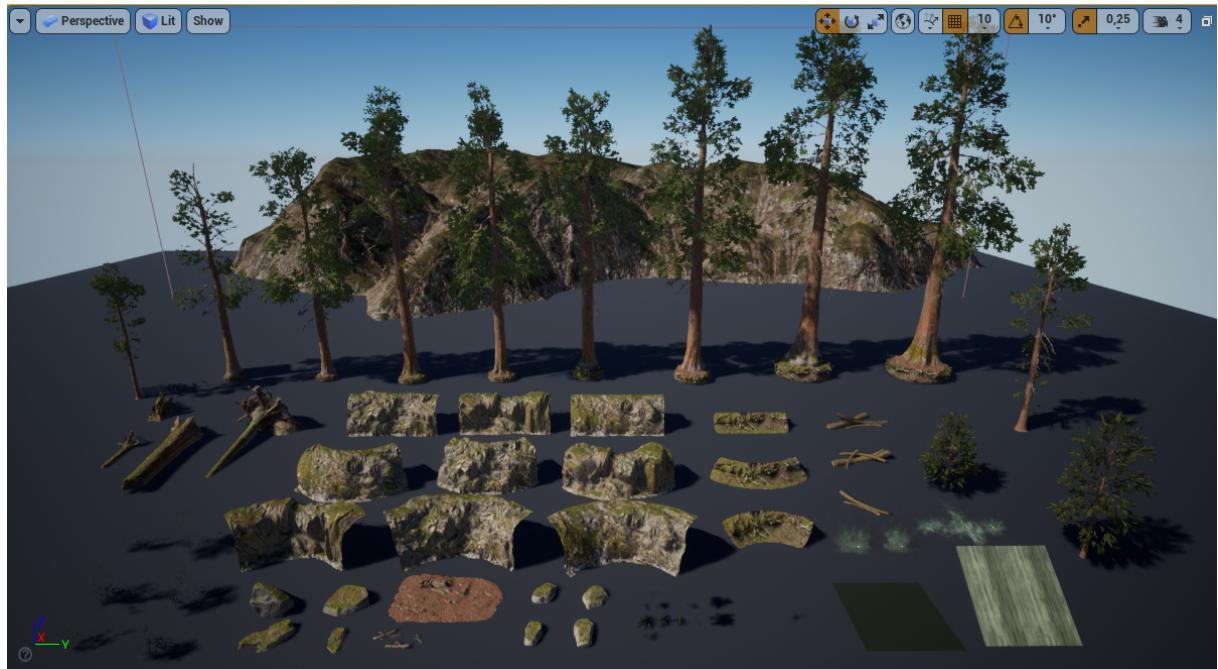
L'environnement "Redwood Forest" fait partie de la famille des mondes dits "procéduriaux". C'est-à-dire que les éléments présents dans le monde n'ont pas été placés à la main par un développeur *designer*<sup>34</sup> mais de manière automatique et procédurale par l'UE4 Editor. Le *designer* n'a alors qu'à produire une quantité limitée de prototypes de chaque objet du monde (arbres, roches, arbustes, petits végétaux, bois mort... etc) et demander à l'éditeur de placer automatiquement un nombre donné de copies de ces prototypes dans le monde. L'intelligence de la génération procédurale réside dans les contraintes de placement des objets dont les positions semblent naturelles à l'oeil humain. Autrement dit, il est très difficile voire impossible de distinguer à l'oeil nu une scène issue d'un monde généré de manière procédurale et la reproduction manuelle d'une scène naturelle.

---

34. Styliste



(a) Prototypes d'arbres



(b) Vue d'ensemble des prototypes

FIGURE 29 – Prototypes d'objets disponibles dans "Redwood Forest"

Dans notre cas, il existe 12 prototypes d'arbres accompagnés de divers prototypes de bois mort, de petits végétaux, de roche et même d'eau pour constituer le bras de rivière qui traverse la carte. Le réseau AdapNet++ étant entraîné à n'identifier que les arbres, j'ai donc commencé par créer une première couleur unie noire et à l'attribuer à tous les objets autres que les arbres (ciel compris), à l'exception du sol auquel j'ai décidé d'attribuer la couleur unie blanche dans l'optique de pouvoir récupérer des informations sur la surface du sol. Couplée aux cartes de profondeur de DenseDepth, cette information pourrait ensuite entrer dans un processus de fusion de donnée afin d'estimer plus précisément les zones d'élévation de terrain. Enfin pour les arbres j'ai créé et attribué 12 autres couleurs unies, une pour chaque prototype.

### Note

J'ai fait le choix de distinguer chaque type d'arbre par une couleur unique pour pouvoir directement entamer un travail sur la segmentation par instance. Pour ce faire, je devais m'assurer de limiter au maximum les cas de chevauchement visuel de plusieurs arbres du même prototype. Cependant je pouvais compter sur l'homogénéité de la distribution des 12 prototypes d'arbres dans le monde "Redwood Forest" grâce à la génération procédurale pour fortement limiter l'observation de chevauchements de troncs identiques.

Le risque de cette stratégie était de fortement augmenter la complexité de la tâche pour le réseau. Cependant, tant que ce dernier réussirait à distinguer les arbres, du sol et des autres éléments, nous aurions au moins eu l'équivalent d'une segmentation sémantique à 3 classes (quand bien même plusieurs classes d'arbres seraient confondues) ce qui aurait été suffisant pour la suite du stage.

Avec cette re-coloration du monde label nous arrivons à un total de 14 classes : une pour le sol, 12 pour les arbres et une pour les autres objets.

Pour finir, les objets relatifs à la lumière de la scène ont été réduits au minimum requis pour obtenir un effet "pastel" le plus neutre possible à l'éclairage du monde.



FIGURE 30 – Vue d'ensemble du monde label.

Malheureusement, le point de neutralité n'aura jamais été complètement atteint en dépit de nos efforts joints avec Charles-Olivier, résultant en des problèmes de cohérence visuelle dans la reconstruction des vues equirectangulaire de nos données. Ces problèmes sont abordés dans la section suivante.

### 3.2.4 Projection equirectangulaire

- Théorie

Une image equirectangulaire est la représentation visuelle d'une scène sphérique dans un espace rectangulaire, ce qui signifie que des opérations de déformation de l'image sont appliquées à celle d'origine. Comme le montre Paul BOURKE sur son excellent site web [8], il existe plusieurs manières de découper une scène d'origine afin de la recomposer en vue equirectangulaire mais l'une des plus communes est de partir d'une représentation dite en *cubemap*<sup>35</sup>. Cette représentation consiste simplement à assembler 6 prises de vue d'une scène en patron de cube de sorte à avoir une description complète de la scène. Le *FOV*<sup>36</sup> de ces prises de vues est souvent défini à 90° pour une meilleure simplicité des calculs.

La transformation d'une vue *cubemap* à une vue equirectangulaire peut ensuite être facilement illustrée de la manière suivante.

35. Littéralement, carte cubique

36. Field Of View : champ de vision

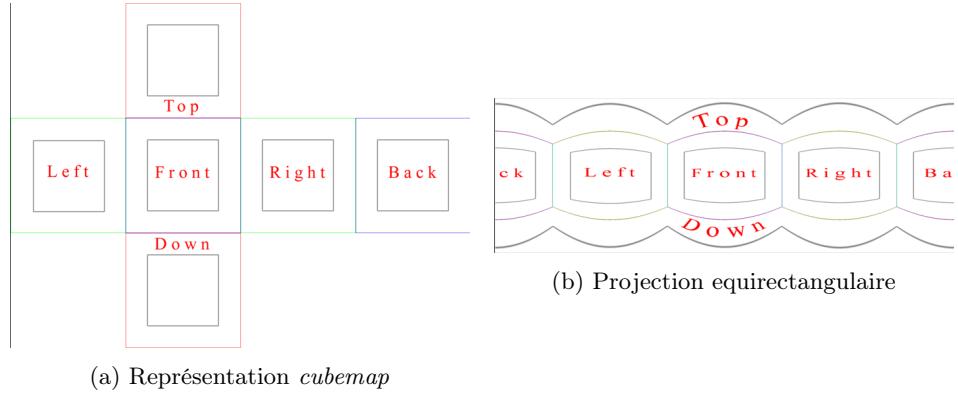


FIGURE 32 – Figures tirées du site de Paul BOURKE [8] illustrant les rapports de géométrie entre les projections *cubemap* et equirectangulaire.

Les conversions géométriques dans le cas des transformations de nos données s'effectuent de la manière suivante.

Une image equirectangulaire de dimensions  $(w, h)$  est considérée exister dans un repère de coordonnées sphériques  $(i, j)$ . Soit  $(x, y)$  le système de coordonnées normalisé sur  $[-1, 1]$ . On a alors :

$$x = \frac{2 * i}{w - 1}, \quad y = \frac{2 * j}{h - 1} \quad (1)$$

Les coordonnées polaires  $(\theta, \phi)$ , bornées respectivement dans  $[0, 2\pi]$  et  $[-\frac{\pi}{2}, \frac{\pi}{2}]$  sont ensuite exprimées en fonction de  $(x, y)$  :

$$\theta = x * \pi, \quad \phi = y * \frac{\pi}{2} \quad (2)$$

Ces coordonnées polaires sont ensuite utilisées pour construire des vecteurs unitaires dans le repère caméra (donc des 6 vues perspective) ce qui permet d'obtenir une formule de passage entre les deux systèmes de coordonnées.

$$x = \cos \phi * \cos \theta, \quad y = \sin \phi, \quad z = \cos \phi * \sin \theta \quad (3)$$

### • Pratique

Une fois la théorie assimilée, j'ai eu la chance de ne pas avoir à développer moi-même d'algorithme la mettant en oeuvre car à cette étape du projet je travaillais conjointement avec Charles-Olivier qui avait déjà mis en place un script à la fois de capture des *cubemaps* (via l'API Python de AirSim) et de conversion en vue equirectangulaire. Cependant ce script était voué à évoluer et nous nous sommes donc coordonnés pour assurer le développement de versions ultérieures du script sur notre dépôt GitHub commun. Ma contribution a principalement été d'apporter de la documentation de code, une re-structuration des méthodes définies ainsi que leur optimisation, l'implémentation de fonctionnalités supplémentaires pour améliorer la praticité d'utilisation et l'ajout de paramétrisation du script.

Le fonctionnement général d'une capture de A à Z est le suivant. Dans un premier temps il faut effectuer un enregistrement via AirSim d'un déplacement manuel dans le monde virtuel. Cela génère un fichier CSV contenant les positions de la caméra à chaque capture. Ce fichier CSV est ensuite augmenté par mon script d'interpolation `augment_airsim_rec.py` (cf. 3.6) pour obtenir plus de positions. Les positions de ce nouveau fichier CSV augmenté sont ensuite lues par notre script de capture des données qui va déplacer automatique la caméra dans le monde virtuel afin de capturer les images perspectives des *cubemaps* dans la modalité demandée (RGB, Segmentation sémantique ou Profondeur). Ces images sont ensuite assemblées et transformées en vue equirectangulaire.

A l'origine, cette transformation était assurée par un appel externe au logiciel de rendu 3D **Blender**, mais à cause de son temps d'exécution particulièrement long, nous avons jugé avec Charles-Olivier qu'il valait mieux trouver une alternative pour accélérer les générations de données. C'est finalement en discutant avec Rida SEKKAT, doctorant au LITIS, que nous avons décidé de remplacer Blender par des *Look Up Table (LUT)*<sup>37</sup>. Les valeurs des matrices de transformation pour passer d'un système de coordonnées

37. Tables de références de valeurs

à un autre ont alors été écrites en dur dans un fichier pour une taille fixe souhaitée. Cela a certes restreint la flexibilité de travail avec les données mais puisque mes réseaux supportent presque tous des entrées de taille variables ce n'était pas un problème.

Ce changement de méthode de transformation nous a permis de diviser par deux le temps nécessaire au script de capture des données de compléter sa chaîne de traitement.

	RGB	SS	Profondeur	Total
Méthode Blender	x	x	x	12s
Méthode LUT	1.6s	2.9s	1.8s	6.3s

TABLE 1 – Test comparatif de la vitesse d'exécution de la chaîne de traitement du script de capture des données pour chaque type de vue (RGB, Segmentation Sémantique (SS), Profondeur).

Partant de ce constat, j'ai fixé mes tailles d'images equirectangulaires à 1000\*500 ce qui me permettait de générer des données dans un temps raisonnable tout en gardant un degré d'information suffisant pour mes réseaux. Pour une base de 785 images capturées à 24fps le temps d'attente était d'environ 3h.

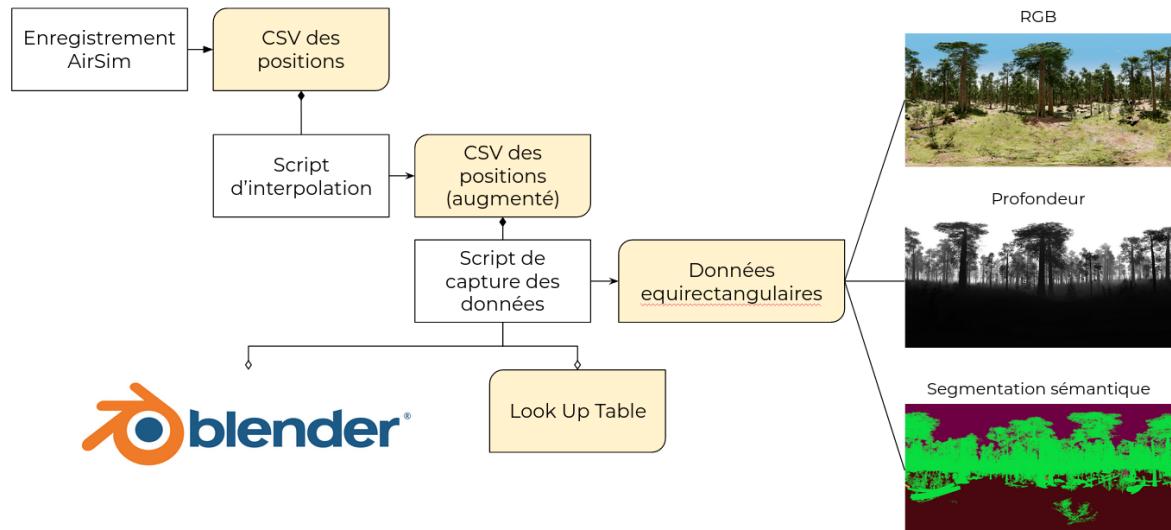


FIGURE 33 – Diagramme de relation lors d'une nouvelle capture complète.

#### • Illumination

Un dernier problème est cependant apparu lors de la reconstitution des images equirectangulaire concernant la gestion de l'illumination de l'image : les vues de dessus et de dessous souffraient d'un fort manque de luminosité. Cette différence n'est pourtant pas visible à l'oeil nu sur les images perspectives des *cube-maps*.



FIGURE 34 – Problème d'illumination sur les vues equirectangulaires de type RGB.

Cela était soudainement très problématique pour les réseaux qui s'entraîneraient alors sur des images biaisées par une luminosité inégale. Charles-Olivier avait déjà rencontré ce problème auparavant sur sa propre copie de "Redwood Forest" et avait réussi à l'éliminer en ajustant les paramètres d'illumination du monde. Dès lors nous avons passé de nouveau en revue les paramétrages d'illumination de mon monde afin de le simplifier au maximum sans pour autant perdre l'aspect visuel photo-réaliste. Mais malgré de multiples vérifications, nous ne sommes pas parvenus à totalement éliminer cet artefact visuel. Ce dernier fut grandement atténué, au point de ne pas pouvoir le distinguer à l'oeil nu sur une vue RGB, mais cela était plus évident dans mon monde label qui souffrait pas conséquent du même problème.

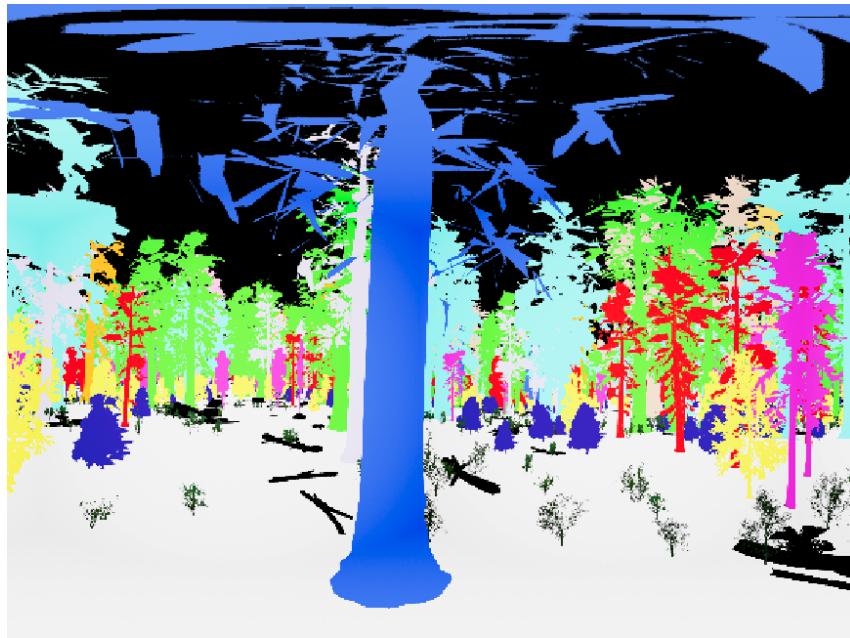


FIGURE 35 – Problème d'illumination sur les vues equirectangulaires de type RGB du monde label. L'artefact visuel est particulièrement visible sur les troncs d'arbres en premier plan.

Au final aucune solution complète et durable ne fut trouvée pour adresser ce problème d'illumination, malgré nos efforts. Pour les vues RGB du monde "Redwood Forest" l'impact était relativement limité et je pouvais espérer que les modèles absorberaient le biais si fortement réduit désormais. Cependant il était impossible de fournir de telles données labellisées car cela aurait signifié que les valeurs des classes ne seraient plus fixes mais constituées d'un gradient de couleur. J'ai donc conçu un algorithme

d'échantillonnage des couleurs par clustering pour adresser ce problème pour les données annotées (cf 3.3.1).

### 3.3 Segmentation sémantique

Dans l'optique de pouvoir proposer un modèle de réseau de neurones capable de réaliser une segmentation sémantique d'arbres sur des images de scènes forestières j'ai effectué du *transfert learning*<sup>38</sup> sur mes données equirectangulaires annotées.

#### 3.3.1 Monde label

Mais afin de pouvoir entraîner mon réseau correctement j'ai d'abord du corriger le problème d'illumination appliquée à mes captures RGB. Pour ne garder que les 12 couleurs unies originellement attribuées à mes prototypes d'arbres, j'ai conçu un script du nom de `cluster_color.py`.

L'approche naïve a d'abord été de comparer chaque pixel de l'image avec les références pour chaque prototypes d'arbres et de redéfinir la couleur du pixel regardé par la couleur référence la plus proche au regard d'une distance choisie. Cependant, ni la distance Euclidienne ni la distance de Manhattan ne semblaient être adaptées à la comparaison de couleurs. J'en ai donc conclu qu'il ne s'agissait pas d'un problème linéaire et qu'il me faudrait un outil plus généralisant.

Je me suis alors tourné vers l'utilisation l'algorithme de *clustering K-means* que j'ai paramétré avec 14 centres de valeurs égales aux 14 couleurs label présentées ci-dessous.



FIGURE 36 – Couleurs unies des prototypes d'arbres. La correspondance des couleurs avec les prototypes est disponible en Annexe 2.

Cette approche fut bien plus pertinente puisqu'elle a permis, à quelques exceptions près, de ré-attribuer correctement des couleurs unies sur la totalité de la surface de chaque arbre. On note tout de même que la couleurs de certains arbres a été inversée mais cela n'est pas dérangeant car ce changement s'est opéré sur l'ensemble des données.

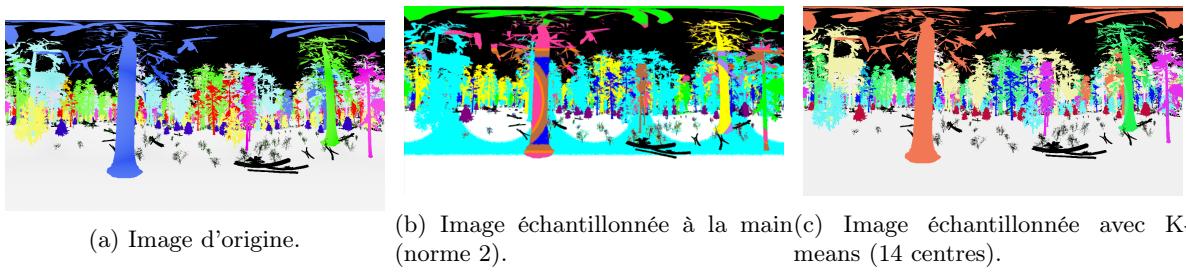


FIGURE 38 – Comparaison des méthodes d'échantillonnage des couleurs des données label.

La complexité de cette opération reste très élevée (4h de calcul pour corriger 785 images de taille 1000 \* 500) mais puisqu'elle n'est censée être lancée qu'une fois ce n'est pas un problème.

#### 3.3.2 AdapNet++

Plus tôt durant ce stage j'ai évalué le réseau AdapNet++ sur des images RGB equirectangulaires issues de "Redwood Forest" mais redimensionnées à la même taille que les données d'entraînement présentées sur le [GitHub de Valada et al.](#). Ceci dans le but d'estimer la robustesse du réseau aux déformations induites par une vue equirectangulaire.

38. Méthode consistant à faire apprendre un modèle avec des poids initialisés sur des valeurs issues d'apprentissages antérieurs (possiblement sur des tâches différentes)

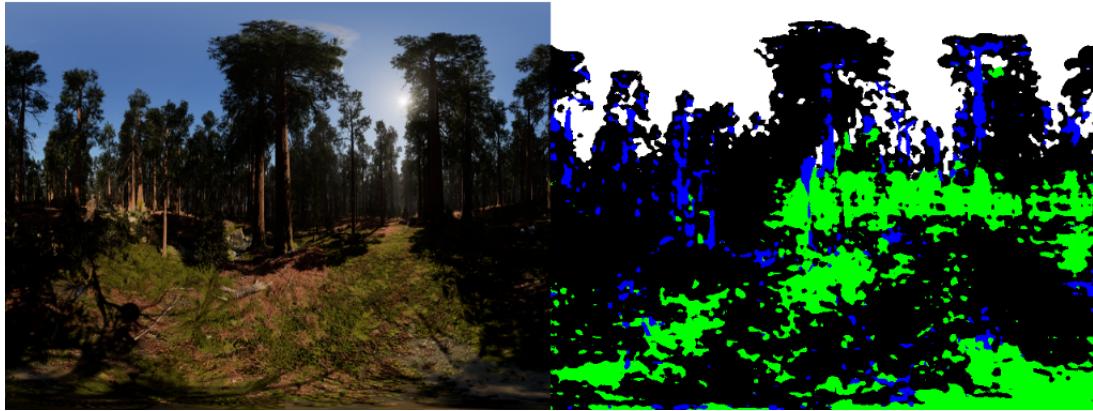


FIGURE 39 – Segmentation sémantique d'AdapNet++ (à droite) sur une vue RGB equirectangulaire de "Redwood Forest" (à gauche).

Les troncs des arbres sont grossièrement identifiés et les feuillages sont distingués comme étant à part entière des arbres, ce qui est normal puisque le modèle a jusqu'ici été entraîné à ne reconnaître que 4 classes : le sol, les troncs, le ciel, les autres éléments. Cependant la précision reste assez faible et il y a beaucoup de zones "faux positifs".

Une fois les données label "nettoyées" j'ai donc ré-entraîné le modèle AdapNet++ sur ces données equirectangulaires en mode supervisé pour voir si le modèle était nativement capable de prendre en compte les déformations liées à cette projection.

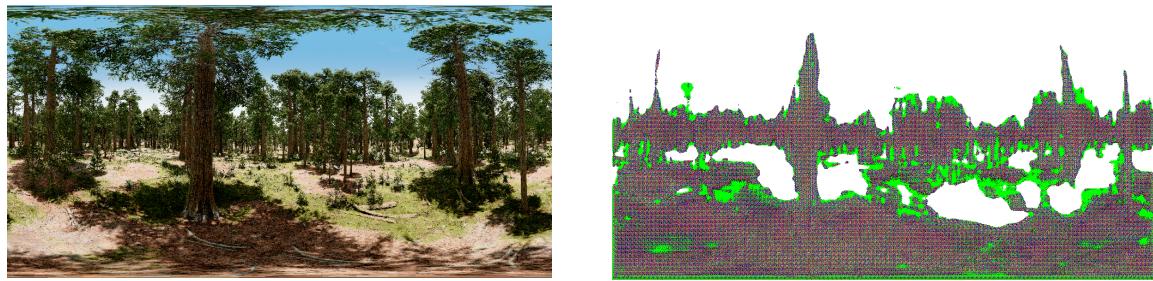


FIGURE 41 – Illustration du problème d'apprentissage d'AdapNet++ sur les données.

On constate immédiatement qu'il y a soit un soucis dans l'apprentissage ou dans les données. Il est naturel de s'attendre à voir plusieurs classes d'arbres mélangées sur le tronc d'un même arbre, mais ici la répartition des classes ressemble plus à un bruit structuré qu'à une erreur de classification. Ce phénomène était déjà apparu au tout début de projet lorsque j'avais évalué ce même modèle, tel que transmis par Dao, sur les données de ce dernier. Une augmentation du nombre d'itération d'entraînement (*epochs*) avait réglé le soucis comme en témoigne la figure 39. Je pose donc l'hypothèse d'un saut d'apprentissage du modèle, mais cela est difficile à confirmer sans outil de visualisation de métriques tels que Tensorboard (difficile à mettre en place sous Tensorflow 1).

Malheureusement faute de temps je n'ai pas pu davantage développer cette partie du projet et la correction du modèle a été ajoutée à la liste des points d'améliorations.

### 3.4 Estimation de profondeur

De manière analogue à la segmentation sémantique, j'ai évalué le modèle DenseDepth pour l'estimation de profondeur sur nos données equirectangulaires, redimensionnées aux dimensions d'entrée par défaut du réseau à savoir  $640 \times 480$ .



FIGURE 43 – Estimation de profondeur par DenseDepth sur une image equirectangulaire redimensionnée.

On constate une plus grande difficulté à estimer correctement les profondeurs qu'avec les données synthétiques de Dao en particulier autour des zones très lumineuses au premier plan qui induisent le réseau en erreur. Encore une fois, sans visualiseur de métriques telles que IoU (*Intersection over Union*) il est difficile de conclure sur la pertinence des prédictions du réseau.

Contrairement à AdapNet++, le réseau DenseDepth n'a pas été ré-entraîné sur nos données equirectangulaires car nous avions préféré avec Pascal creuser la piste d'un autre modèle déjà pré-entraîné sur des données déformées : le 360SD-Net.

#### 3.4.1 360SD-Net

Ce modèle *deep learning* développé par Ning-Hsu Wang et al. [9], totalisant 5,306,215 paramètres est un réseau de neurones spécialement conçu pour la prise en compte de déformations visuelles engendrées par les projections equirectangulaires. Bien qu'entraîné sur des *datasets* de scènes d'intérieur tels que **MP3D** ou **SF3D**, nous avions parié sur la possibilité de réussir un *transfert learning* étant donné la qualité des prédictions du réseau sur ses *datasets* de prédilection [10].

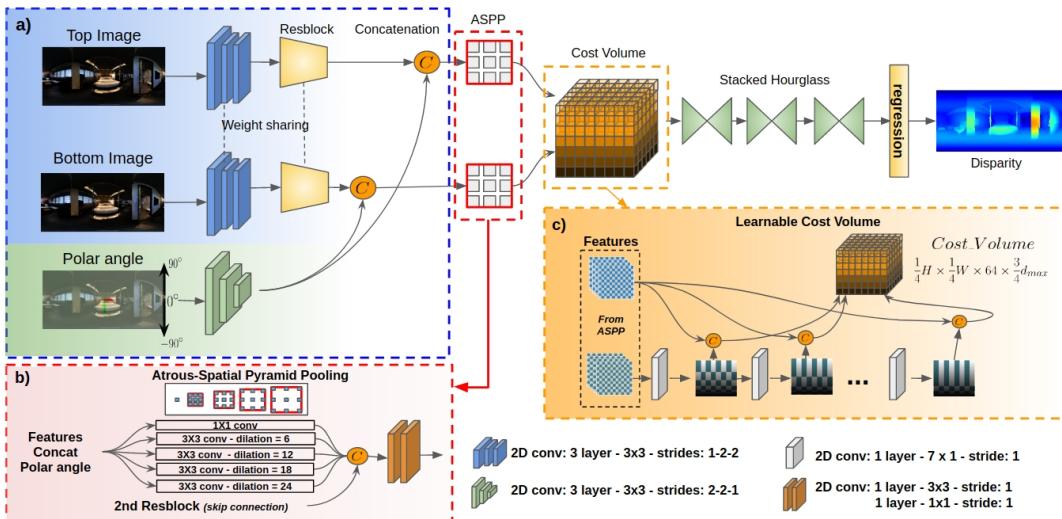


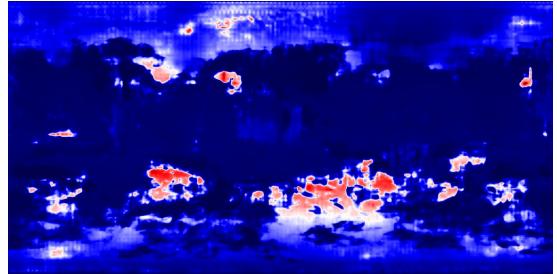
FIGURE 44 – Architecture du modèle 360SD-Net.

Grâce à son module *ASPP (Atrous-Spatial Pyramid Pooling)*, le réseau 360SD-Net est en mesure de prendre en compte les déformations d'une image equirectangulaire et utilisant des couches de convolution à noyau éparse, de manière à caractériser l'éloignement des pixels sous forme de *features*<sup>39</sup>. Ces *features* sont ensuite balayées dans le module *LCV (Learnable Cost Volume)* par différents filtres de taille  $7 * 1$  dans l'objectif de trouver la déformation optimale pour chaque portion  $7 * 1$  de ces *features*. Ceci permet par la suite de définir une matrice de coût volumique dont les poids sont utilisés par un *Encoder-Decoder* chargé de reconstruire une carte de disparité précise et adaptée aux déformations de la projection equirectangulaire.

Malheureusement pour nous, il s'est avéré que le réseau éprouvait beaucoup de mal à comprendre la géométrie de nos scènes forestières au point où les prédictions étaient inexploitables. Des faiblesses similaires ont par ailleurs également été constatées sur des exemples d'intérieur equirectangulaires récupérés d'internet. Mais n'ayant pas eu le temps nécessaire à allouer à l'approfondissement d'une solution adaptative de ce modèle, nous avons laissé de côté cette tâche pour la reporter elle aussi à la liste des évolutions du projet.



(a) Image RGB d'origine



(b) Prédiction de 360SD-Net

FIGURE 46 – Estimation de profondeur par 360SD-Net sur une image equirectangulaire de notre jeu de données.

### 3.5 Crédation de cartes

Beaucoup de temps de travail a été dégagé pour mener à bien cette partie cartographie car elle représentait la phase de travail avec le plus d'incertitudes et pour cause : tout était à faire. L'objectif premier que nous nous sommes fixé était de réussir à générer des cartes de localisation 2D avec à minima la position des arbres repérés à proximité du drone ainsi qu'une estimation du diamètre de ces arbres.

Les aspérités du terrain, les cas particuliers d'arbres non droits et autres cas complexe ont été mis de côté pour s'assurer de posséder une bonne base. Etant donné que les résultats de nos réseaux de neurones était approximatifs pour certains, et très mauvais pour d'autres, nous sommes partis dans l'optique de nous baser sur les vérités terrains offertes par AirSim et Unreal Engine 4 pour développer notre méthode de cartographie.

#### 3.5.1 Vérité terrain

Les données de vérité terrain qu'il nous fallait récupérer étaient :

1. Les positions des arbres
2. Les envergures de chaque prototype d'arbre

Les envergures étaient directement mesurables via l'UE4 Editor et n'ont pas posé de problème pour être obtenus.

39. Caractérisation d'une information dans un réseau de neurone

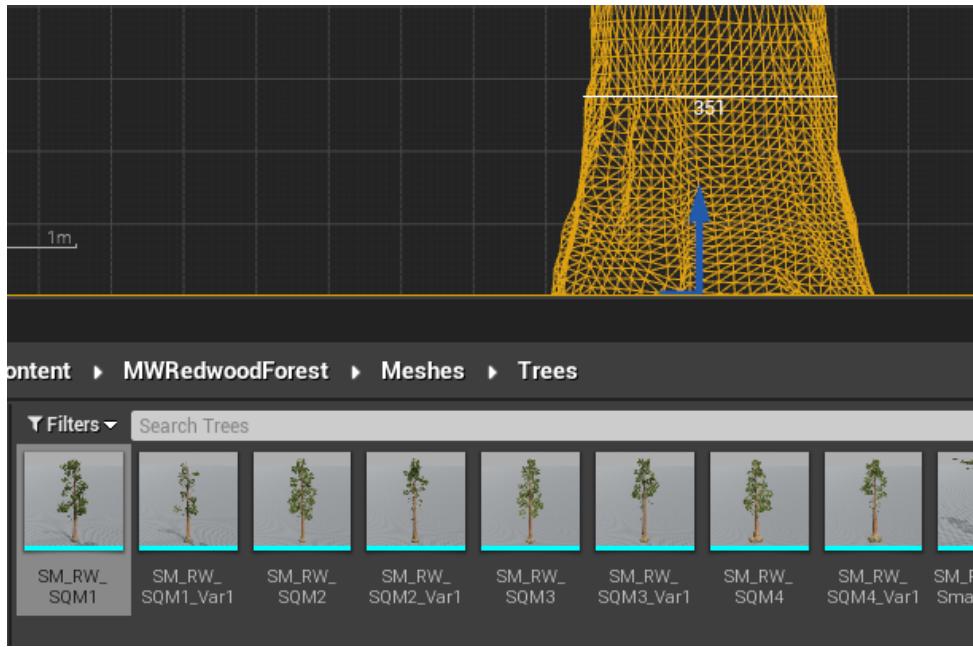


FIGURE 47 – Mesure de l'envergure du tronc d'un des prototypes d'arbres sous UE4 Editor.

Name	Girth (m)
SM_RW_SQM1	0.350
SM_RW_SQM1_Var1	0.350
SM_RW_SQM2	0.410
SM_RW_SQM2_Var1	0.410
SM_RW_SQM3	0.520
SM_RW_SQM3_Var1	0.450
SM_RW_SQM4	0.735
SM_RW_SQM4_Var1	0.725
SM_RW_SQMSmall1_Var1	0.065
SM_RW_SQMSmall1_Var2	0.090
SM_RW_SQMSmall2_Var1	0.025
SM_RW_SQMSmall2_Var2	0.015

TABLE 2 – Tableau de recensement des envergures des prototypes d'arbres en mètres.

En revanche, récupérer les positions des arbres a demandé beaucoup plus de recherche. Les arbres étant générés de manière procédurale, ils existent de manière individuelle avec un ID de génération mais ne sont sélectionnables individuellement dans l'UE4 editor. L'accès à leur informations n'est pas non plus triviale via l'API de AirSim. A terme, j'ai réussi à formuler une solution sous la forme d'un *Blueprint*<sup>40</sup> grâce à l'aide d'un membre de la communauté du forum Unreal. Cette solution a par ailleurs été doublement utile puisque Charles-Olivier cherchait en vain un moyen de répondre à ce problème également.

40. Langage de programmation visuel d'Unreal Engine

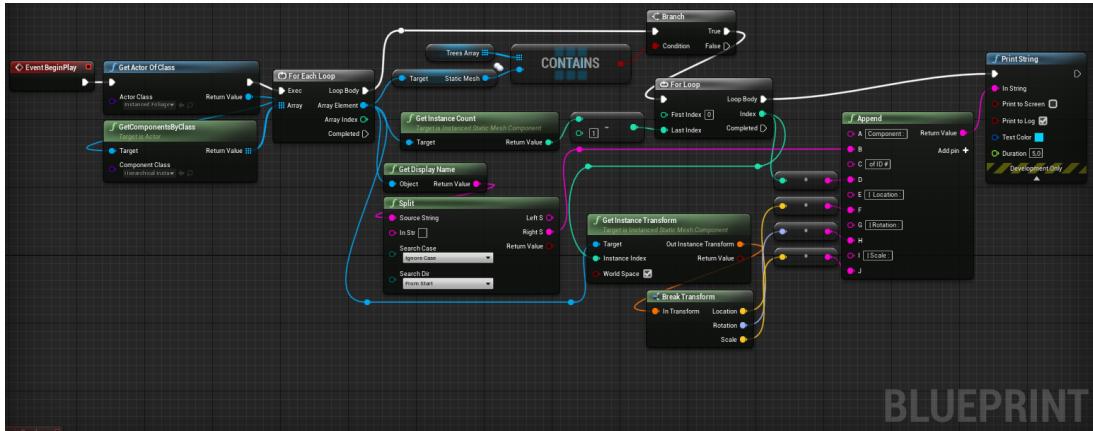


FIGURE 48 – Blueprint solution pour récupérer les informations de positions d’arbres instanciés (image également disponible en Annexe 3).

Toutes les informations de vérité terrain ont été renseignées dans des fichiers CSV pour être par la suite utilisées par l’algorithme de cartographie.

### 3.5.2 Carte 2D

Lors de l’élaboration de la carte, était d’imaginer un système de fusion de données de sorte que l’on puisse déterminer pour chaque arbre, sa distance au drone et son envergure à partir de vues RGB, Segmentation sémantiques et Profondeurs.

La solution que nous avons imaginée avec Pascal se base sur le principe de **Bitterlich - Relaskop** [11], une méthode d’estimation du nombre d’arbres dans une région donnée en se basant sur leur distance à l’observateur et leur diamètre. L’observateur tient à bout de bras un appareil de mesure optique gradué, le Relaskop, et regarde chaque arbre autour de lui à travers une lentille. Si l’envergure de ces arbres remplit le champ de vision visible à travers la lentille, l’arbre est compté comme existant dans la région autour de l’observateur. Dans le cas contraire, l’arbre est ignoré. Il est ensuite possible de déterminer la concentration en arbres dans la zone en multipliant le nombre d’arbres sélectionnés par une constante de densité d’aire  $K$  appelée *Basal areal density*.

$$K = \left( \frac{50 * B}{L} \right)^2 \quad (4)$$

Avec quelques connaissances à priori sur les variables  $R$ ,  $B$  et  $L$  il est également possible de retrouver le diamètre approximatif des arbres sélectionnés par une simple règle de proportionnalité (Thalès). C’est de ce principe que nous nous sommes inspirés pour construire notre méthode.

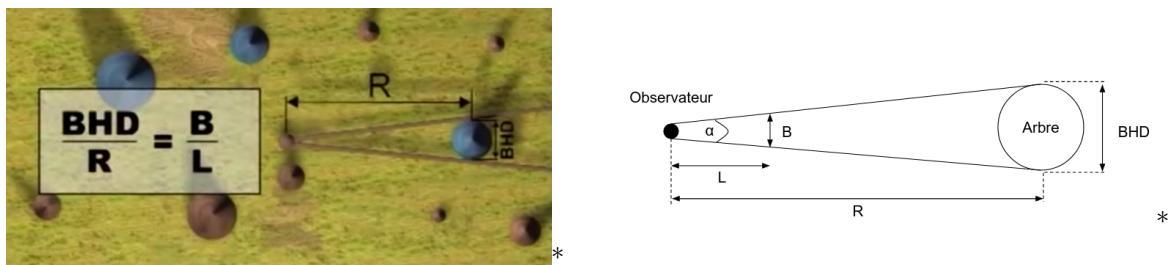


FIGURE 50 – Principe de bitterlich.

Sources : Youtube [12]

Les données utilisées pour construire notre carte sont les cartes de profondeur et les segmentations sémantique (abrégée SS). Les vues RGB quant à elles ne sont utiles que dans le cadre d’une démonstration visuelle.

Pour chaque image, on va venir dans un premier temps quadriller la vue SS, d'un nombre de lignes et de colonnes défini par l'utilisateur de manière à préparer le terrain pour un échantillonnage prochain. Nous nous reposons sur l'heuristique du cas simple où le drone qui a capturé les images ne s'est ni trop incliné en avant ni trop incliné en arrière ce qui a pour effet d'offrir des images d'arbres dont les troncs sont toujours visibles autour du centre vertical de l'image. De ce quadrillage, on ne vient donc retenir que les 5 lignes les plus proches du centre vertical de l'image.

Puis, en se servant de la carte de profondeur, la SS désormais réduite à ces 5 lignes de quadrillage centrales, va être échantillonnée dans sa profondeur à la manière d'un accordéon. Les valeurs de la carte de profondeur vont quant à elles jouer le rôle de masques binaires. Le fonctionnement par défaut de l'algorithme lui fait sélectionner 5 intervalles de valeur égaux entre 0 et 100 (ou toute autre valeur spécifiée par l'utilisateur), correspondant aux bornes minimales et maximales des profondeurs à prendre en compte. Nous aurons alors les intervalles suivants : [0, 20], [20, 40], [40, 60], [60, 80], [80, 100]. Ce sont ces intervalles qui vont servir à définir les masques de profondeur que l'on va venir appliquer sur l'image SS. Nous nous retrouvons donc avec une matrice à 3 dimensions de largeur  $w$ , hauteur 5 et profondeur  $d$  comportant, pour chaque strate  $d_i$ , les valeurs de l'image SS pour les pixels dont les positions correspondent avec ceux de la carte de profondeur dont les valeurs sont dans l'intervalle de la strate en question.

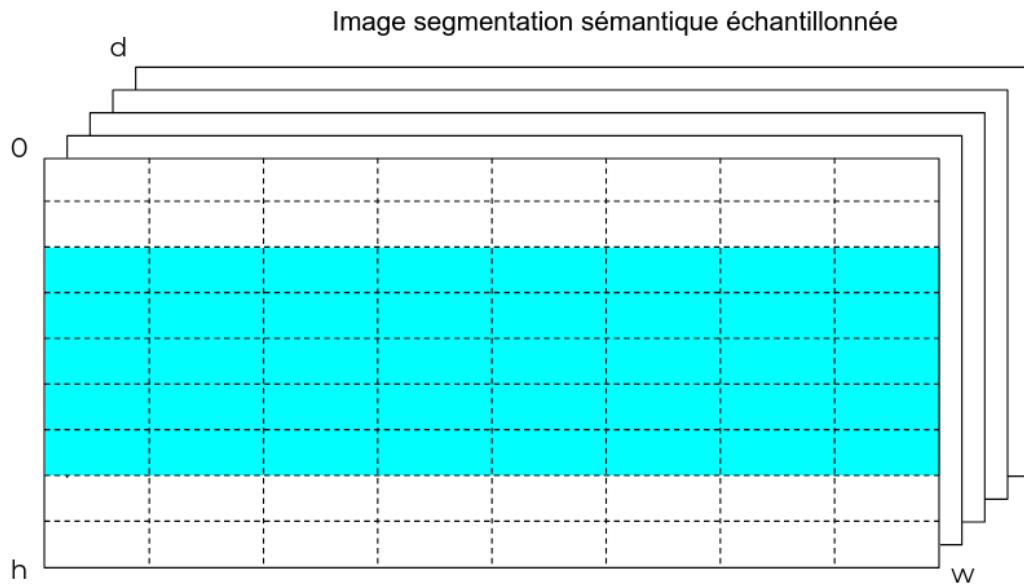


FIGURE 51 – Quadrillage de l'image SS sur sa hauteur et largeur ; et échantillonnage sur sa profondeur.

Pour une meilleure compréhension, voici comment visualiser chaque découpage dans la profondeur ou "strate".

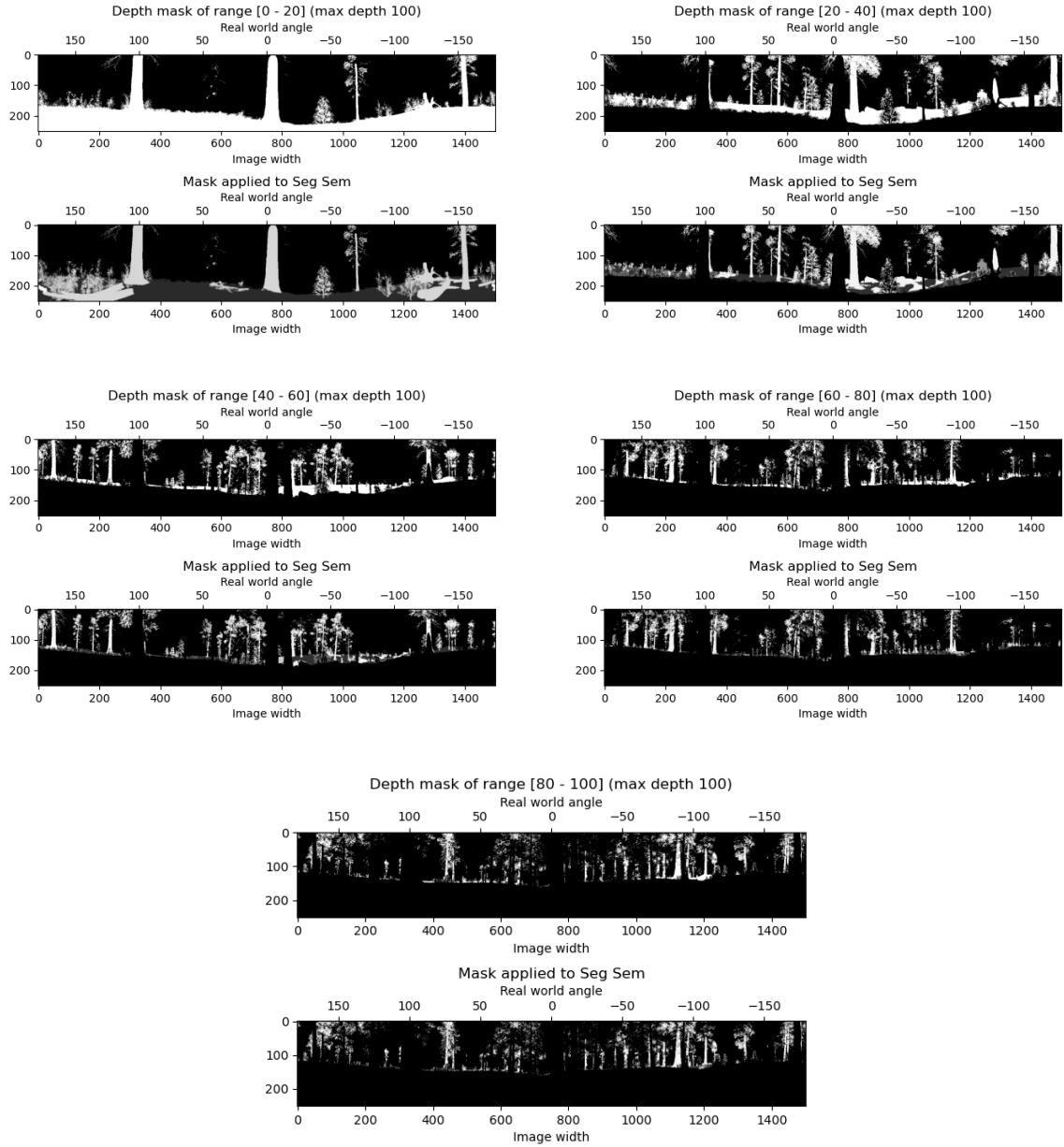


FIGURE 53 – Découpage en profondeur des masques et de l'image SS en niveaux de gris.

A présent l'algorithme va venir scanner chacune des lignes du quadrillage, qui contiennent encore des pixels de l'image SS. Chaque cellule du repère quadrillé, sera désormais représenté par un nombre flottant borné de 0 à 1 caractérisant la proportion de pixel d'arbre dans la cellule. Une absence d'arbre résultera en un 0. Une cellule ne comptant que des pixels d'arbres sera codée par un 1.

Notons  $e$  le nombre de cellule dans un ligne (qui correspond également au nombre de colonnes du quadrillage). Notre nouvelle matrice est maintenant de taille  $(e, 5, d)$  avec  $d = 5$  par défaut, et ses valeurs sont une représentation simplifiée et plus légère de la présence d'arbre à chaque intervalle de profondeur.

Désormais il ne reste plus qu'à décider quels valeurs de la matrice appartiennent à un arbre. Pour cela, la matrice est effondrée sur sa hauteur par opérateur de moyenne ou de médiane, au choix. Désormais de taille  $(e, 1, d)$  nous cherchons à identifier dans chaque strate une succession de valeurs flottantes suffisamment élevées et successives qui identifieraient la présence d'un tronc. Un seuil minimal empirique de 0.2 est défini pour pouvoir considérer une valeur comme faisant partie d'un tronc. Puis en balayant

les vecteurs  $e, 1, i$  pour  $i$  allant de 1 à  $d$ , va déclencher la découverte d'un tronc lorsque plusieurs valeurs successives supérieures ou égales au seuil minimal sont lues. Ces valeurs, ainsi que la position de leur découverte dans le vecteur, sont stockées. A l'issue des balayages, l'algorithme va venir retrouver la profondeur exacte de chaque cellule en effectuant un moyennage sur la section équivalente en pixels de la carte de profondeur.

Enfin pour retrouver la position par rapport au drone de chaque arbre identifiés, l'algorithme va appliquer une règle de proportionnalité similaire à celle présentée dans la méthode de Bitterlich. La différence étant que dans la configuration de Bitterlich, notre variable  $L$  est nulle puisque l'acquisition est faite par une caméra, il nous faut donc générer une échelle à partir de nos SS. La manière la plus simple est de calculer, pour chaque cellule de profondeur estimée  $d_e$ , le périmètre autour de la caméra (qui correspond à la longueur totale de nos images equirectangulaires) pour un rayon  $r = d_e$ . Connaissant le nombre de cellules  $e$  dans notre matrice, il est facile d'établir un rapport cellules/pixels/angle.

Nous avons donc à ce stade récupéré, pour chaque identification d'arbre, sa distance et son orientation relative à la caméra (représentative du drone). Il ne reste plus qu'à afficher nos prédictions sur notre carte 2D. Pour ce faire, j'ai choisi d'utiliser la librairie *matplotlib* en itérant sur chaque image equirectangulaire de notre base de données. La carte est toujours centrée sur le drone dont la trace est affichée par des croix bleus. Les points rouges représentent la vérité terrain des arbres tandis que les points bleus représentent les prédictions de l'algorithme. Chaque arbre prédit est enfin relié à l'arbre de la vérité terrain le plus proche.

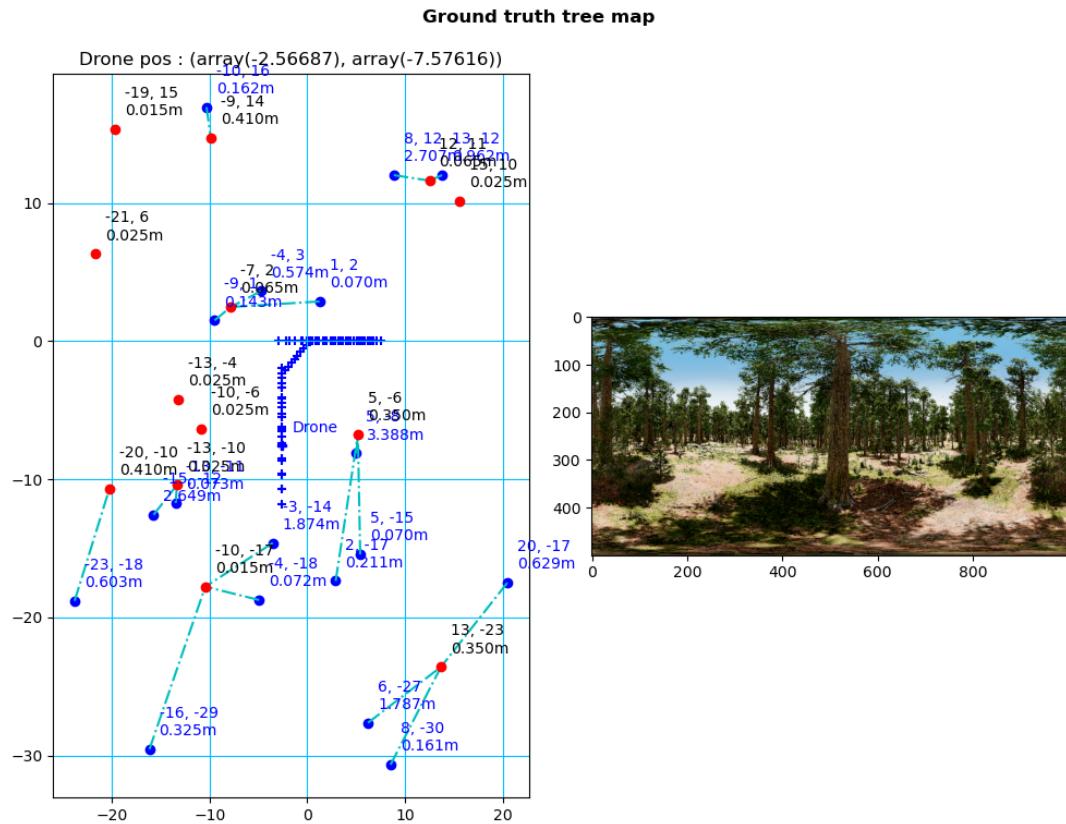


FIGURE 54 – Affichage en temps réel de l'estimation des arbres (bleu) VS la vérité terrain (rouge).

### 3.5.3 Résultats

Nous avons pu quantifier la précision de nos prédictions par le biais des métriques *Mean*<sup>41</sup>, *Median*<sup>42</sup> et *RMSE*<sup>43</sup> dont les résultats sont les suivants.

41. Moyenne

42. Médiane

43. *Root Mean Square Error* : racine de l'erreur des moindres carrés

Métrique	Angle (°)	Distance (m)	Envergure (m)
Mean	93.82	0.69	2.54
Median	121.67	0.35	2.15
RMSE	589.77	8.46	17.04

TABLE 3 – Résultats de notre solution selon les métriques Moyenne, Médiane et RMSE

De plus nous avons défini des scores de *Precision*<sup>44</sup> et de *Recall*<sup>45</sup> en considérant comme valide, tout arbre dont la vérité terrain la plus proche est à une distance égale à la moyenne de ces distances à +/- 15%.

Métrique	Valeur
Precision	0.593
Recall	0.067

TABLE 4 – Estimation des métriques *Precision* et *Recall* pour une validation des *True Positive* à flexibilité 15%.

On remarque tout d’abord que la vaste majorité des prédictions est très proche d’une vérité terrain puisque l’erreur de distance est fortement minimisée. On pourrait pour autant garder un oeil critique sur cette métrique en considérant que la densité d’arbre de la vérité terrain pourrait être suffisamment forte pour qu’il y ait toujours au moins 1 arbre proche des prédictions, mais le chemin choisi par le drone a été spécialement choisi pour passer par des espaces à densités d’arbres différentes. Notamment, le drone est passé par deux reprises au milieu d’une grande clairière où très peu de troncs d’arbres sont présents mais où l’on rencontre des arbustes qui tendent à déclencher de fausses détection. Pour cette raison, nous pensons que cette métrique est légitime et témoigne d’une bonne localisation des arbres.

En revanche, l’estimation des envergures est autrement moins précise. D’après la vérité terrain, tous les diamètres d’arbres sont inférieurs au mètre, or la médiane des envergures estimées dépasse déjà les 2 mètres. Cette partie de l’algorithme serait à revoir car il ne s’agit en l’état pas de résultats exploitables.

Concernant l’erreur sur les angles, je pense qu’il réside un problème dans le calcul de cette métrique car en pratique, les prédictions sont correctement placées sur la carte. Mais je n’exclus pas la possibilité d’un décalage ou d’un effet miroir sur la position des points à cause d’un mauvais calcul d’angle dans le repère cartésien. Cependant je n’ai pas réussi à trouver la source de cette anomalie avant la fin de mon stage.

Pour finir, les métriques de *Precision* et de *Recall* sont à nuancer car elles ont été basées sur un seuil arbitraire afin de pouvoir exister. Néanmoins nous pouvons émettre une première conjecture. Au vu des scores de ces deux métriques et de ce que l’on peut observer sur la carte, il semblerait que parmi les prédictions de l’algorithme beaucoup sont justes, mais qu’il resterait encore beaucoup d’arbres non détectés.

### 3.5.4 Dimension temporelle

Le dernier point de développement manquant était désormais l’intégration de la dimension temporelle pour la correction de prédictions douteuses, en se basant par exemple sur la stabilité dans le temps des arbres estimés. Tant qu’un arbre reste dans un périmètre autour du drone d’un rayon inférieur à celui de la distance maximale, il est censé apparaître sur la carte pour toute capture. Cependant, pour des raisons évoquées en conclusion 1 et par manque de temps, nous n’avons pas pu proposer de méthode fonctionnelle intégrant les méthodologies SLAM à ce projet.

Nous avons cependant pu tester l’algorithme ORB-SLAM 3 [13] sur nos données equirectangulaires. Bien que les matrices de paramétrisation des caméras d’ORB-SLAM3 ne correspondaient pas tout à fait à nos projections equirectangulaires, en approximant les coefficients polynomiaux de distorsion des modèles de caméras nous avons tout de même eu la preuve qu’il était possible de récupérer des descripteurs persistants sur nos données, pourvu que la distance entre chaque image consécutive ne soit pas trop grande.

44. Précision

45. Rappel



FIGURE 55 – Accroche de descripteurs visuels (carrés verts sur la fenêtre de droite) avec ORB-SLAM3 à nos données equirectangulaires et création d'un nuage de point (points rouges sur la fenêtre de gauche) des éléments proches.

### 3.6 Scripts divers

De manière complémentaire à toutes les productions présentées précédemment, j'ai également pris le temps de développer un arsenal de scripts-outils paramétrables qui m'a été indispensable à tout niveau du projet et qui peut selon moi être tout aussi utiles pour de futurs développeurs.

- **Redimensionnement d'images**

Le script `reshape_img.py` m'a permis de pouvoir redimensionner à la volée une ou plusieurs images présentes dans un dossier de manière récursive à la taille voulue en utilisant la librairie OpenCV avec une interpolation de type bilinéaire. Ce script a été particulièrement utile pour ajuster la taille des images d'entrée des réseaux de neurones, notamment pour les images de vraies forêts récupérées d'internet. Des tests unitaires ont été développés pour assurer son bon fonctionnement.

- **Parser<sup>46</sup> de positions**

Afin de pouvoir exploiter plus facilement les positions extraites sous format texte du monde "Redwood Forest" j'ai développé un script *parser* pour exporter chaque information dans un tableau CSV. Le script se base sur des heuristiques de formatage du texte et sur des règles de grammaire pour séparer les champs. Ce formatage des données m'a servi pour la construction des cartes 2D. Des tests unitaires ont été développés pour assurer le bon fonctionnement du script.

Données brutes :

```
LogBlueprintUserMessages: [BP_Collector_C_0] Component : SM_RW_SQM1_Var1 of
ID #0 / Location : X=-42240.137 Y=14968.323 Z=1592.129 / Rotation : P=1.70
7745 Y=69.937828 R=-1.041112 | Scale : X=0.800 Y=0.800 Z=0.800
LogBlueprintUserMessages: [BP_Collector_C_0] Component : SM_RW_SQM1_Var1 of
ID #1 / Location : X=-40335.176 Y=33854.078 Z=1113.656 / Rotation : P=-1.5
21595 Y=77.351288 R=-1.298131 | Scale : X=0.800 Y=0.800 Z=0.800
LogBlueprintUserMessages: [BP_Collector_C_0] Component : SM_RW_SQM1_Var1 of
```

46. Analyseur syntaxique

ID #2 / Location : X=-26291.498 Y=-23067.824 Z=852.795 / Rotation : P=-0.4  
 37112 Y=-163.422653 R=0.056842 | Scale : X=0.800 Y=0.800 Z=0.800

Données formatées :

Name	ID	LocX	LocY	LocZ	RotP	RotY	RotR	ScaX	ScaY	ScaZ
SM_RW_SQM1_Var1	0.0	-422.40137	149.68323	15.92129	1.707745	69.937828	-1.041112	0.8	0.8	0.8
SM_RW_SQM1_Var1	1.0	-403.35176	338.54078	11.13656	-1.521595	77.351288	-1.298131	0.8	0.8	0.8
SM_RW_SQM1_Var1	2.0	-262.91498	-230.67824	8.52795	-0.437112	-163.422653	0.056842	0.8	0.8	0.8

FIGURE 56 – Données de positions des arbres formatées en CSV.

#### • Interpolation de positions

Mon utilisation principale du *plugin* AirSim a été de lancer des enregistrements de scènes dans le monde "Redwood Forest" et le monde label afin de récupérer des positions de caméra sous forme de fichier CSV. Cependant, les performances *hardware* de ma machine ont conduites à une limitation de la fréquence maximale de capture à 4 *fps*<sup>47</sup>. Cela a introduit des problèmes de continuité entre les scènes capturées car les sauts entre chacune d'entre elles étaient trop importants, rendant très difficile leur utilisation par mes algorithmes de création de carte et celui d'ORB-SLAM3. De plus, cela rendait compliqué la compréhension visuelle lorsque l'on jouait les images les unes à la suite des autres.

C'est pourquoi j'ai développé le script `augment_airsim_rec.py` qui s'est assuré d'augmenter le nombre de captures en insérant de nouvelles lignes de positions à l'intérieur du fichier CSV, interpolées à partir des existantes. Cela a eu pour effet d'augmenter la fréquence de capture des *cubemaps* et donc de réduire la distance entre les images equirectangulaires de mes bases de données. Des tests unitaires ont été développés pour assurer le bon fonctionnement de ce script.

#### • Edition de fichiers image et CSV

Lors de l'utilisation de ORB-SLAM3 j'ai construit un panel de scripts destiné à manipuler des images ainsi que le contenu de fichiers CSV (contenant des références de fichiers et d'ID de capture).

Le script `rename_files_from_csv.py`, par exemple, m'a permis de renommer toute une batterie d'images en fonction des valeurs de certaines cellules d'un fichier CSV comportant des références de capture.

Un autre script `reduce_csv.py` m'a permis de redimensionner mes CSV à références en fonction du nombre d'images présentes dans mon dossier de données. Cela a été particulièrement pratique lorsqu'il a fallu effectuer différentes captures dans le monde "Redwood Forest" à des fréquences de capture variées.

D'autres scripts `blur_imgs.py` et `to_grayscale.py` appliquent, comme leur nom l'indique, des opérations de flou et de mise en niveaux de gris sur une batterie d'images spécifiée. L'utilité de ces scripts s'est faite ressentir pour tester la robustesse d'ORB-SLAM3 sur des images plus ou moins nettes et riches en information.

47. *Frame Per Second* : trame par seconde

# Troisième partie

## Conclusion

### 1 Appréciation

Ce stage ingénieur aura été pour moi l'occasion de me mettre dans la peau d'un ingénieur recherche travaillant sur des sujets complexes et pluri-disciplinaires. Ce dernier point est sans doute celui ayant constitué la principale difficulté d'avancement dans le projet car cela m'a demandé de me former en semi ou complète autonomie à la manipulation de nouveaux projets existants et solutions logicielles - en particulier Unreal Engine 4 Editor, Visual Studio et AirSim - ce qui a constitué un challenge certain au vu du temps restreint dont je disposais pour conduire mon stage (5 mois).

Cela a demandé une organisation particulière pour assurer la parallélisation des tâches prioritaires afin de ne pas laisser des problématiques clefs sans réponses. Cependant il s'est avéré que ma stratégie n'a pas su prouver son efficacité sur le long terme car arrivé en fin de projet, certaines tâches importantes restaient encore non adressées tandis que d'autres moins prioritaires devaient être conclues car avaient déjà été significativement entamées. Avec le recul, je pense qu'il aurait été bénéfique d'instaurer un système de classement par priorité des tâches à effectuer ainsi que de posséder un tableau d'avancement des objectifs, similaire aux outils utilisés en méthodologie Scrum, Kanban etc... En contrepartie, cela aurait signifié d'accepter de sacrifier un peu de temps au début du projet pour mettre en place ces outils, mais cet effort aurait certainement été largement récompensé à terme.

Malgré des difficultés éprouvées à maintenir le rythme en fin de projet (cf. 2) j'ai cependant été satisfait de pouvoir conduire ce projet en complète autonomie dans un environnement de travail propice à la discussion scientifique. En effet, contrairement à mes précédentes expériences professionnelles, le laboratoire MIS abrite de nombreux doctorants, chercheurs et maîtres de conférence auprès de qui il est facile de demander conseil. Je pense même qu'il est important de chercher à croiser les expertises car à plusieurs reprises cela m'a aidé à sortir d'une "vision tunnel" dans laquelle je pouvais par moment basculer à force d'acharnement, en pointant de nouvelles pistes aboutissant à de nouvelles solutions.

Ensuite j'ai pu pour la première fois prendre part activement à des réunions scientifiques sous la forme de séminaires ayant tantôt pour but de rendre compte d'un avancement des tâches dans un projet donné, tantôt pour assister à la présentation de l'oeuvre complète d'autres doctorants ou stagiaires. Cela fut toujours hautement instructif et a parfois permis d'établir un dialogue avec des acteurs travaillant sur des problématiques différentes mais néanmoins connexes aux miennes, de sorte que l'on s'apporte mutuellement de l'aide. J'ai notamment travaillé conjointement avec Charles-Olivier sur les aspects géométrie lors de la création des vues equirectangulaires et nous avons chacun pu apporter des solutions et conseils précieux à l'autre. Cette opération d'échange de connaissance étant au cœur de la recherche scientifique de nos jours, je suis très content d'avoir pu l'expérimenter dans le cadre de ce stage.

Enfin, j'ai pu prendre véritablement conscience durant ces 5 mois de l'importance des enjeux psychologiques relatifs à la conduite autonome d'un projet à grand potentiel dans des conditions sociales diminuées. Il n'est pas sage de sous-estimer l'importance d'une hygiène mentale saine pour la bonne réussite d'un projet autonome, qui plus est dans des circonstances de distanciation sociale liées à la crise du COVID-19 puisqu'il n'y a personne sur qui se reposer lorsque le besoin se fait sentir. Je suis certain, à l'avenir, de prévoir par avance des contre-mesures efficaces pour enrayer ce phénomène de démoralisation.

Au final le projet est tout de même désormais doté d'une base de solution pour la cartographie dynamique des arbres en environnement forestier ainsi que de plusieurs briques annexes et les prochains objectifs sont d'ores et déjà fixés, ce qui s'avèrera très utile en cas de reprise de mes travaux. Pour ma part, malgré une petite déception de ne pas avoir pu atteindre plus d'objectifs, je reste satisfait de mon stage dont je saurai assurément tirer profit des enseignements qu'il m'aura procuré. Je reste également convaincu par le domaine de la recherche scientifique et ce stage aura été une formidable occasion d'échanger avec des acteurs du milieu de la thèse doctorale (directeurs de thèse, doctorants, enseignants-chercheurs...).

## 2 Critiques et difficultés

La première difficulté à laquelle je ne m'attendais pas a été le délai de prise en main des nouveaux outils et nouvelles ressources. Il est bien plus difficile de reprendre un projet que d'en commencer un et le temps nécessaire à la compréhension en détail des travaux de Dao a été d'autant plus important que nos échanges ne pouvaient dépasser un rythme de un à deux mails par jour en raison du décalage horaire important entre nous. Il en est allé de même avec les autres ressources telles que l'environnement UE4 Redwood Forest, où des prises de contact avec les concepteurs du monde ainsi que des membres du forum UE4 Hub ont été nécessaires, mais également avec l'éditeur UE4 Editor et le plugin AirSim qui m'ont collectivement demandé de nombreuses semaines d'acclimatation avant de pouvoir me sentir à l'aise avec.

Au regard de ce frein au développement, je pense que ma stratégie de répartition des tâches aurait du être adaptée bien plus tôt dans le projet afin de garder plus de marge de manœuvre en fin de stage où les tâches essentielles auraient alors déjà été clôturées, quitte à "tricher" en contournant certaines phases de développement nécessaires en amont en fournissant artificiellement les sorties attendues.

D'autre part, le déroulement de ce stage aura été marqué par les restrictions sanitaires liées à la crise de COVID-19 nous forçant à limiter les présences et contacts entre collègues sur le lieu de travail. Bien que nous ayons déjà tous goûté aux méthodes de télétravail depuis l'an dernier, j'ai trouvé difficile le fait de supporter, sur la durée, les effets de l'isolement social compte tenu du fait que je travaillais seul sur le projet et que mes relations sociales fréquentes étaient dépendantes de la présence de mes camarades stagiaires et doctorants au laboratoire, en même temps que la mienne. Les effets de la perte de moral se sont surtout fait sentir sur la fin du stage lorsque la quasi totalité du personnel est partie en congés et que nous n'étions plus que deux dans le bâtiment ; mes camarades ayant tous déjà terminé leur stage. Les conséquences ont été une perte significative de vitesse d'avancement sur la fin du projet menant à des retards sur certains points clefs jusque-là mis en attente.

## 3 Travail restant et évolution

### 3.1 Objectifs en attente

La prise en compte des aspérités du terrain a été relayée en composante "optionnelle" peu de temps après le commencement du projet pour être finalement mise de côté au vu du rythme d'avancement du projet mais s'avèrera indispensable pour la réalisation du projet ANR CLARA dans sa totalité.

Il en va de même pour la détection des cas particuliers d'arbres couchés ou courbes qui peuvent barrer la route du drone. La solution actuelle se basant sur des euristiques de verticalité pour repérer les troncs, elle n'est pas en mesure d'assurer la détection de ces dangers.

### 3.2 Points d'évolution

La prise en compte de la dimension temporelle reste le gros point à compléter dans ce projet pour deux raisons. La première est qu'on observe un certain nombre de détections d'arbres vacillantes d'une image à une autre, c'est-à-dire des prédictions qui apparaissent et disparaissent sur des images pourtant très similaires car très proches dans le temps. La prise en compte de leur détection dans le temps permettrait de déterminer leur pertinence en tentant de retrouver chaque détection à différents moments de capture. Cela permettrait de mieux définir les groupes *TP* (*True Positive*) et *FN* (*False Negative*) voire même de fusionner plusieurs détections très proches spatialement afin d'atténuer les erreurs de prédictions liées à des cassures dans les images post-traitement.

Un autre point important serait d'arriver à des réseaux de neurones capables de comprendre à la fois les déformations induites par les images equirectangulaires et la structuration des forêts de manière à fournir des cartes de profondeur et des segmentations sémantiques pertinentes qui sont jusqu'à présent balbutiantes.

L'ajout de la dimension "hauteur" dans les cartes sera également, à terme, un aspect à développer, en particulier pour s'adapter aux forêts en milieu montagneux.

## Références

- [1] Abhinav VALADA et al. « AdapNet : Adaptive Semantic Segmentation in Adverse Environmental Conditions ». In : *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, p. 4644-4651.
- [2] Ibraheem ALHASHIM et Peter WONKA. « High Quality Monocular Depth Estimation via Transfer Learning ». In : *arXiv e-prints* abs/1812.11941, arXiv :1812.11941 (2018). eprint : [1812.11941](https://arxiv.org/abs/1812.11941). URL : <https://arxiv.org/abs/1812.11941>.
- [3] Brad NEMIRE. NVIDIA Researchers Release Trailblazing Deep Learning-Based Framework for UAV. URL : <https://developer.nvidia.com/blog/nvidia-researchers-release-trailblazing-deep-learning-based-framework-for-autonomous-drone-navigation/> (visité le 03/04/2021).
- [4] Barbara Webb JAMES GARFORTH. *Visual Appearance Analysis of Forest Scenes for Monocular SLAM*. 12 août 2019. URL : [https://ieeexplore.ieee.org/abstract/document/8793771?casa\\_token=kdwcKPuZyb8AAAAA:8fYnMyuxoEjroaVJYh8L3M7iYuQ\\_A6x29LiV9pCUN7sec5B73bv-US6MAT1GlaQ7gSA9i5P1EGwvBA](https://ieeexplore.ieee.org/abstract/document/8793771?casa_token=kdwcKPuZyb8AAAAA:8fYnMyuxoEjroaVJYh8L3M7iYuQ_A6x29LiV9pCUN7sec5B73bv-US6MAT1GlaQ7gSA9i5P1EGwvBA).
- [5] *Unreal : Panoramic Capture Tool*. URL : <https://docs.unrealengine.com/4.26/en-US/WorkingWithMedia/StereoPanoramicCapture/> (visité le 22/03/2021).
- [6] Prarthana Shresth et AL. *Synchronization of Multiple Camera Videos Using Audio-Visual Features*. 1<sup>er</sup> jan. 2010. URL : <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5332301> (visité le 05/04/2021).
- [7] *Git Airsim*. URL : <https://github.com/microsoft/AirSim> (visité le 08/04/2021).
- [8] Paul BOURKE. *Converting to/from cubemaps*. Nov. 2020. URL : <http://paulbourke.net/panorama/cubemaps/> (visité le 01/06/2021).
- [9] Ning-Hsu Wang et AL. *360SD-Net : 360° Stereo Depth Estimation with Learnable Cost Volume*. 11 nov. 2019. URL : <https://arxiv.org/abs/1911.04460>.
- [10] Ning-Hsu Wang et AL. *Git 360SD-Net*. URL : <https://github.com/albert100121/360SD-Net> (visité le 30/03/2021).
- [11] László Czúni et AL. *Color Based Clustering for Trunk Segmentation*. 20 août 2018. URL : [https://ieeexplore.ieee.org/abstract/document/8439358?casa\\_token=PdlcQ8rqu44AAAAA:e00U1YqTjarKfx8wJx1ShQdtKGZX1UNDNq\\_Vpp6kV1SHu5MFrW00nx9VA15qwkWDD9v0TKP4-mJVQ](https://ieeexplore.ieee.org/abstract/document/8439358?casa_token=PdlcQ8rqu44AAAAA:e00U1YqTjarKfx8wJx1ShQdtKGZX1UNDNq_Vpp6kV1SHu5MFrW00nx9VA15qwkWDD9v0TKP4-mJVQ).
- [12] *Youtube : Bitterlich Method - Relaskop*. 26 jan. 2016. URL : <https://www.youtube.com/watch?v=VggNOuWfEtQ> (visité le 17/05/2021).
- [13] Carlos Campos et AL. *ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM*. 23 avr. 2021. URL : <https://arxiv.org/pdf/2007.11898.pdf> (visité le 01/06/2021).
- [14] Steven W. Chen et AL. *SLOAM Semantic Lidar Odometry and Mapping for Forest Inventory*. 29 déc. 2019. URL : <https://arxiv.org/abs/1912.12726>.
- [15] Jiazheng Liu et AL. *Classification of tree species and stock volume estimation in ground forest images using Deep Learning*. Nov. 2019. URL : [https://www.sciencedirect.com/science/article/pii/S0168169919308713?casa\\_token=9osOumrLLMwAAAAA:SVPt3-jMNWhU\\_0milazh\\_L1ZLBzBVWSjfGAmJml7CDrQEL\\_zBx4uY8yMcnz0ZD9XuX7fpNMNKM](https://www.sciencedirect.com/science/article/pii/S0168169919308713?casa_token=9osOumrLLMwAAAAA:SVPt3-jMNWhU_0milazh_L1ZLBzBVWSjfGAmJml7CDrQEL_zBx4uY8yMcnz0ZD9XuX7fpNMNKM).
- [16] Bert De Brabandere et AL. *Semantic Instance Segmentation with a Discriminative Loss Function*. 8 août 2017. URL : <https://arxiv.org/abs/1708.02551>.
- [17] Carlos Campos et AL. *Semantic Instance Segmentation with a Discriminative Loss Function*. 12 mar. 2021. URL : <https://arxiv.org/pdf/2003.05766.pdf> (visité le 01/06/2021).
- [18] Thomas SHARPLESS. *Synchronizing Digital Stereo Cameras*. URL : <https://ivrpa.org/forums/topic/exposure-sync-with-digital-stereo-cameras/> (visité le 05/04/2021).
- [19] *Genlock*. URL : <https://en.wikipedia.org/wiki/Genlock> (visité le 05/04/2021).
- [20] *The NerdCam3D Mk.1 - Your Versatile Stereoscopic FPV Flight Camera*. URL : <https://www.themissinggear.eu/nerdcam3d-mk-1/> (visité le 05/04/2021).
- [21] *Collection of 360° Video Rigs*. URL : <https://thefulldomeblog.com/2015/11/17/collection-of-360-video-rigs/> (visité le 05/04/2021).

- [22] *Git ORB-SLAM3*. URL : [https://github.com/UZ-SLAMLab/ORB\\_SLAM3](https://github.com/UZ-SLAMLab/ORB_SLAM3) (visité le 04/06/2021).
- [23] MAUHING. *Git ORB-SLAM3 (aide installation)*. URL : [https://github.com/Mauhing/ORB\\_SLAM3/blob/master/README.md](https://github.com/Mauhing/ORB_SLAM3/blob/master/README.md) (visité le 04/06/2021).
- [24] *UnrealCV documentation*. URL : <https://unrealcv.org/> (visité le 22/03/2021).
- [25] *Airsim documentation*. URL : <https://microsoft.github.io/AirSim/apis/> (visité le 08/04/2021).
- [26] *OpenCV documentation*. URL : <https://docs.opencv.org/master/> (visité le 04/06/2021).
- [27] Théo LARCHER. *UE4 forum : blueprint solution*. 14 juin 2021. URL : <https://answers.unrealengine.com/questions/1035455/how-to-get-position-of-procedural-foliage.html?childToView=1037188> (visité le 24/06/2021).
- [28] *Unreal : A comprehensive guide to creating 360-degree game trailers using Unreal*. 6 fév. 2020. URL : <https://www.unrealengine.com/en-US/tech-blog/a-comprehensive-guide-to-creating-360-degree-game-trailers-using-unreal> (visité le 22/03/2021).
- [29] *Youtube : Render 360 VR videos out of Unreal Engine*. 24 déc. 2020. URL : <https://www.youtube.com/watch?v=x4orAa0mrBo> (visité le 22/03/2021).
- [30] *Stackexchange : How to convert a normal photo into an equirectangular image ?* 7 fév. 2017. URL : <https://graphicdesign.stackexchange.com/questions/84774/how-to-convert-a-normal-photo-into-an-equirectangular-image> (visité le 01/06/2021).

## Table des figures

2	Ville d'Amiens	2
3	Antennes de l'UPJV	3
5	Ville d'Amiens	4
6	Outil d'agenda en ligne GRR	5
8	Composants principaux	8
10	Comparaison des vues de profondeur de la scène n°24 issue du monde virtuel UE4.16 de Dao	9
12	Comparaison des vues segmentation sémantique de la scène n°24 issue du monde virtuel UE4.16 de Dao	10
13	<i>Clustering</i> de troncs isolés de la scène n°24 du monde virtuel UE4.16 de Dao	11
17	Exemples de DenseDepth sur des images de scènes réelles (à gauche les images de référence ; à droite les prédictions du réseau)	12
19	Exemples de AdapNet++ sur des images de scènes réelles (à gauche les images de référence ; à droite les prédictions du réseau)	13
20	Monde UE4.16 de Dao où les textures des arbres ont été modifiées pour rendre les troncs et feuillages d'une couleur unie	13
21	Monde UE4.26 Redwood Forest	15
23	Exemple comparatif des deux types de vue. Ici, la vue perspective correspond au centre de la vue equirectangulaire.	16
24	Schéma extrait de [6] démontrant l'intérêt de baser le recalage des flux vidéos sur un décalage initial.	19
25	Editeur de monde d'Unreal Engine 4.26 (exemple sur "Redwood Forest")	20
26	Simulation de drone sous AirSim.	21
27	Monde label issu du monde "Redwood Forest".	22
29	Prototypes d'objets disponibles dans "Redwood Forest"	23
30	Vue d'ensemble du monde label.	24
32	Figures tirées du site de Paul BOURKE [8] illustrant les rapports de géométrie entre les projections <i>cubemap</i> et equirectangulaire.	25
33	Diagramme de relation lors d'une nouvelle capture complète.	26
34	Problème d'illumination sur les vues equirectangulaires de type RGB.	27
35	Problème d'illumination sur les vues equirectangulaires de type RGB du monde label. L'artefact visuel est particulièrement visible sur les troncs d'arbres en premier plan.	27
36	Couleurs unies des prototypes d'arbres. La correspondance des couleurs avec les prototypes est disponible en Annexe 2.	28
38	Comparaison des méthodes d'échantillonnage des couleurs des données label.	28
39	Segmentation sémantique d'AdapNet++ (à droite) sur une vue RGB equirectangulaire de "Redwood Forest" (à gauche).	29
41	Illustration du problème d'apprentissage d'AdapNet++ sur les données.	29
43	Estimation de profondeur par DenseDepth sur une image equirectangulaire redimensionnée.	30
44	Architecture du modèle 360SD-Net.	30
46	Estimation de profondeur par 360SD-Net sur une image equirectangulaire de notre jeu de données.	31
47	Mesure de l'envergure du tronc d'un des prototypes d'arbres sous UE4 Editor.	32
48	<i>Blueprint</i> solution pour récupérer les informations de positions d'arbres instanciés (image également disponible en Annexe 3).	33
50	Principe de bitterlich.	33
51	Quadrillage de l'image SS sur sa hauteur et largeur ; et échantillonnage sur sa profondeur.	34
53	Découpage en profondeur des masques et de l'image SS en niveaux de gris.	35
54	Affichage en temps réel de l'estimation des arbres (bleu) VS la vérité terrain (rouge).	36
55	Accroche de descripteurs visuels (carrés verts sur la fenêtre de droite) avec ORB-SLAM3 à nos données equirectangulaires et création d'un nuage de point (points rouges sur la fenêtre de gauche) des éléments proches.	38
56	Données de positions des arbres formatées en CSV.	39
57	Roadmap issue du rapport de Dao mettant en schéma les pistes qu'il a explorées	46
58	Correspondance des couleurs label avec les prototypes d'arbres.	47
59	Solution <i>Blueprint</i> pour récupérer les vérités terrain sous l'UE4 Editor.	48

## Liste des tableaux

1	Test comparatif de la vitesse d'exécution de la chaîne de traitement du script de capture des données pour chaque type de vue (RGB, Segmentation Sémantique (SS), Profondeur). . . . .	26
2	Tableau de recensement des envergures des prototypes d'arbres en mètres. . . . .	32
3	Résultats de notre solution selon les métriques Moyenne, Médiane et RMSE . . . . .	37
4	Estimation des métriques <i>Precision</i> et <i>Recall</i> pour une validation des <i>True Positive</i> à flexibilité 15%. . . . .	37

## Annexes

## A Annexe 1

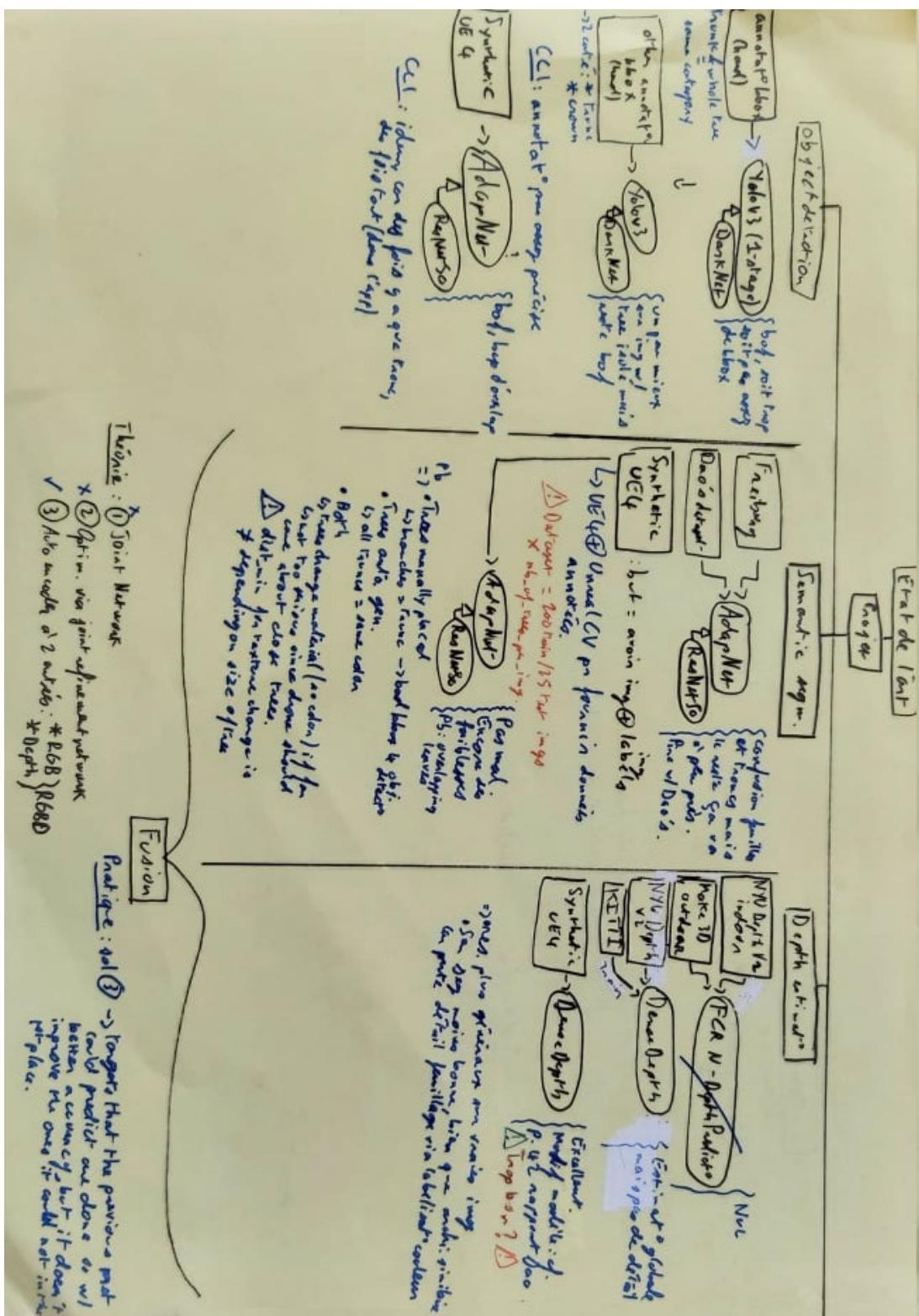


FIGURE 57 – Roadmap issue du rapport de Dao mettant en schéma les pistes qu'il a explorées

## B Annexe 2

Component_ID	Color_ID	R	G	B	H	S	V	Gray
SM_RW_SQM1	SolidBlue	0.0	0.0	255.0	240.0	1.0	1.0	76
SM_RW_SQM1_Var1	SolidBlue2	164.2274499999999	134.14071	255.0	254.931442	0.473958	1.0	174
SM_RW_SQM2	SolidCyan	0.0	255.0	255.0	180.0	1.0	1.0	226
SM_RW_SQM2_Var1	SolidGreen	0.0	255.0	0.0	120.0	1.0	1.0	150
SM_RW_SQM3	SolidGreen2	184.66929	107.881065	64.624905	21.63089	0.649937	0.723958	103
SM_RW_SQM3_Var1	SolidOrange	255.0	60.77619	0.0	14.300293	1.0	1.0	64
SM_RW_SQM4	SolidOrange2	255.0	131.36376	14.804025000000001	29.11615	0.941945	1.0	110
SM_RW_SQM4_Var1	SolidPink	250.92	65.34375	156.974175	330.374298	0.739583	0.984	113
SM_RW_SQMsmall11_Var1	SolidPurple	127.5	0.0	127.5	300.0	1.0	0.5	52
SM_RW_SQMsmall11_Var2	SolidRed	152.73021	0.0935849999999999	0.0	0.036774	1.0	0.598942	17
SM_RW_SQMsmall12_Var1	SolidYellow	255.0	255.0	0.0	60.0	1.0	1.0	179
SM_RW_SQMsmall12_Var2	SolidDarkBlue	7.69998	0.0	58.437585000000006	247.905853	1.0	0.229167	18
None	SolidRed2	34.533350000000006	0.0	0.0	0.0	1.0	0.135417	4
Ground	SolidWhite	255.0	255.0	0.0	0.0	0.0	1.0	255
Other	SolidBlack	0.0	0.0	0.0	0.0	0.0	0.0	0

FIGURE 58 – Correspondance des couleurs label avec les prototypes d'arbres.

## C Annexe 3

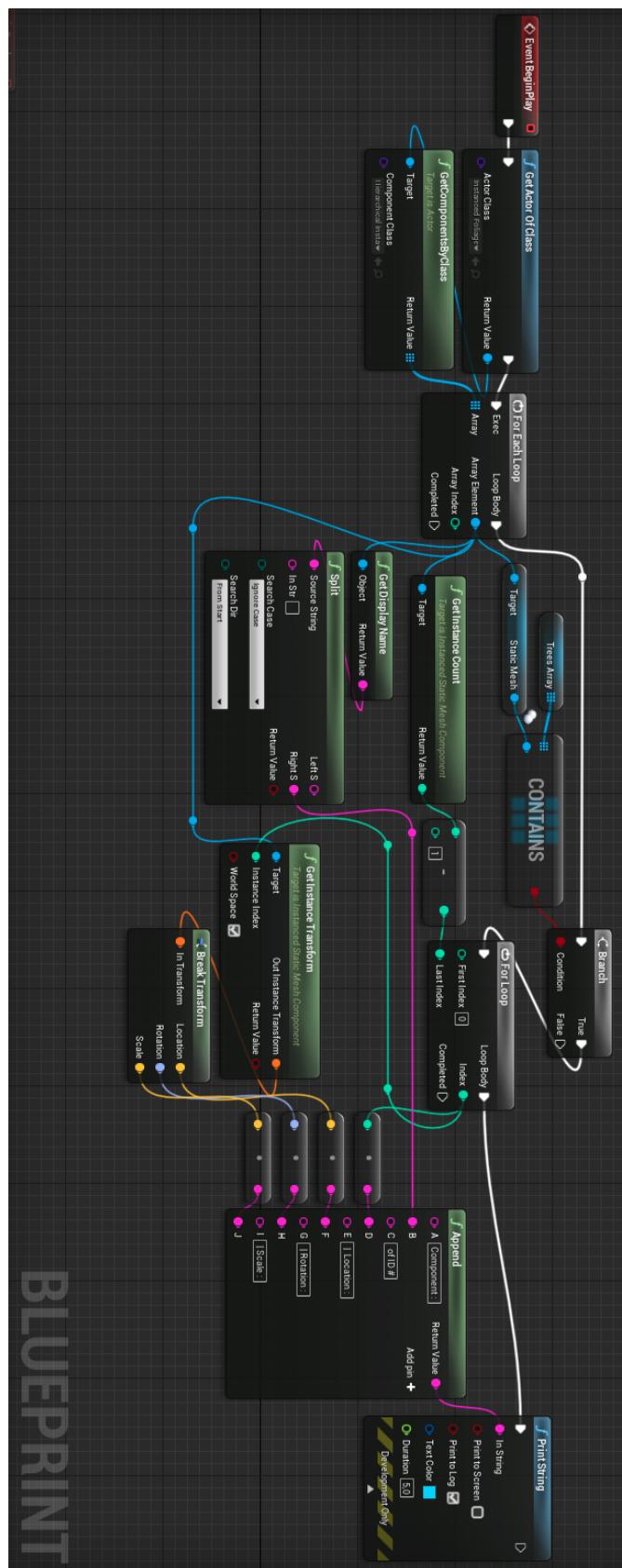


FIGURE 59 – Solution Blueprint pour récupérer les vérités terrain sous l'UE4 Editor.

Ce rapport présente mon travail effectué durant mon stage ingénieur INSA opéré au laboratoire *MIS* d'Amiens pour le compte du laboratoire LITIS à Saint-Etienne du Rouvray et sous la tutelle de Pascal VASSEUR dans le cadre du projet ANR CLARA. Ce stage orienté recherche est la reprise d'un travail précédemment amorcé par un ancien stagiaire, Dao ZHOU, et est une contribution scientifique à la fois pour le laboratoire MIS mais aussi pour le laboratoire I3S de Nice puisque c'est sous le signe de la co-opération inter-équipes que s'est déroulé ce stage.

A base de réunions hebdomadaires, de partage de connaissances et de participation à des séminaires de laboratoire, ce stage aura été une véritable opportunité de se projeter dans la vie quotidienne des ingénieurs recherche, des doctorants et des enseignants chercheurs. Venant de tous les coins du monde, la diversité culturelle au sein des équipes de travail a été mise à l'honneur, démontrant que le savoir scientifique est une notion qui se travaille et se partage sans barrières ni distinction d'âge ou d'origine.

Dans ce rapport nous aborderons des aspects techniques concernant la vision par ordinateur appliquée à l'imagerie omnidirectionnelle pour la génération automatique des cartes forestières. De la paramétrisation des caméras aux modèles de deep learning destinés à identifier des arbres dans des images en passant par la génération de données synthétique, nous aborderons toutes les étapes d'un projet de recherche condensé en 5 mois.

Marqué par la crise du COVID-19 et les mesures d'hygiène et de distanciation social, nous reviendrons sur l'adaptation nécessaire à ce mode de travail sous contraintes et des conséquences que cela peut avoir sur le développement d'un projet de recherche.

Enfin, la méthodologie de travail, les choix des solutions techniques et une auto-critique sur la gestion du projet durant ce stage occuperont également une place proéminente dans ce rapport. En effet, bien que ce stage ne fut ni le plus aisé, ni le plus couronné de succès, il fut cependant indéniablement formateur au rôle d'ingénieur de recherche. Il est donc important de capitaliser sur cette expérience.

This report presents my 5<sup>th</sup> year internship's work carried out at the *MIS* laboratory of Amiens, 80000 France on behalf of the LITIS laboratory of Saint-Etienne du Rouvray, 76800 France under the direction of Pascal VASSEUR as part of the ANR CLARA project.

This internship which is strongly research oriented, is a resumption of the work of former intern Dao ZHOU and represents a scientific contribution both benefiting the MIS lab, but also the I3S laboratory of Nice, 06000 France, since this internship has been devoted to cross-team cooperation.

Animated with weekly meetings, knowledge sharing and seminars participation as well as animation, this internship will have truly been a great opportunity to set foot in the daily life of research engineers, PhD students and university lecturer and researcher. Coming from all around the globe, cultural diversity among teams has definitely been under the spotlight, thus demonstrating that scientific research knows no frontiers.

In this report we will tackle several technical aspects regarding computer vision applied to omnidirectional imaging for automated forest mapping. From camera parametrization to deep learning models designed to identify trees in sceneries by way of synthetic data generation, we will go over every step of a research project, condensed in 5 months.

Scarred by the COVID-19 crisis inducing health related measures and social distancing, we will go over the necessary adaptation for this constrained work model and analyse its consequences on the development of a research project.

Lastly, the chosen work strategy and technical solutions will be among the topics we will be dealing with here, along with a critic of our own work. Indeed, even if this experience hasn't been the easiest nor the most successful, it was definitely educational about the role of a research engineer. This is a work experience I most certainly will benefit from.

## INSA Rouen Normandie

685, Avenue de l'Université  
76800 Saint-Étienne-du-Rouvray - France  
[www.insa-rouen.fr](http://www.insa-rouen.fr)



MINISTÈRE  
DE L'ÉDUCATION NATIONALE,  
DE L'ENSEIGNEMENT SUPÉRIEUR  
ET DE LA RECHERCHE