



Projet ObsHyper

Rapport final

Emilie SIMON

Jany LAHLOUH

Julien DUCHEMANN

Pierre-Olivier PEDRENO

Résumé

ObsHyper est un projet ayant pour ambition de créer un système radar fonctionnel à partir des différents éléments constituant le standard de n'importe quel radar. Il s'agira donc d'apprehender la théorie relative à l'univers des systèmes d'observations, dont une certaine partie est intégrée à notre cursus, d'apprehender les différents organes des-dit systèmes, ainsi que de mettre en pratique cette théorie pour programmer différents algorithmes capables de mettre en œuvres différents assemblages radars. Une fois les prémisses d'un système fonctionnel obtenues, il sera pertinent de suggérer des améliorations à celui-ci telles que le suivi de cible, le traitement "temps réel", ou encore la complexité de nos algorithmes.

Lors de la première partie du projet nous avions compris le rôle et le fonctionnement de chacune des parties d'un radar, et avons pu les mettre en œuvre individuellement et collectivement, toutefois nous avons été interrompus par le confinement et manquons d'expérience pratique. Du point de vue théorique, nous avons mis au point deux algorithmes, l'un traitant le signal reçu par le radar et l'autre capable d'isoler le bruit du signal utile.

A la suite de quoi, nous avons affiné nos algorithmes et nous les avons reliés pour former une chaîne fonctionnelle capable de détecter, suivre et rapporter une cible à un ensemble déjà construit. Pour ce faire, nous avons effectué des mesures et des acquisitions en extérieur, sur une cible mobile.

Mots clés : RADAR, acquisition, traitement, signal, suivi de cible.

Abstract

OBSHyper is a project with the ambition to create a functional radar system from the different elements that make up the standard of any radar. It will thus be a question of apprehending the theory relating to the universe of observation systems, a certain part of which is integrated into our curriculum, of apprehending the various organs of these systems, as well as putting this theory into practice in order to program various algorithms capable of implementing different radar assemblies. Once the beginnings of a functional system are obtained, it will be relevant to suggest improvements to this system such as target tracking, real-time processing, or the complexity of our algorithms.

So far, we have understood the role and operation of each part of a radar and have been able to implement them individually and collectively, however we have been interrupted by the lockdown and we suffer from a lack practical experience. From a theoretical point of view, we have developed two algorithms, one processing the signal received by the radar and the other able to isolate the noise from the wanted signal.

Following this, we refined our algorithms and linked them to form a functional chain capable of detecting, tracking and reporting a target to an already built set. To do this, we performed measurements and acquisitions outdoors, on a moving target.

Key words: RADAR, acquirement, processing, signal, tracking.

Remerciements

Nous souhaitons, tous les quatre, remercier nos encadrants le Dr. Fabrice Comblet et M. Didier Tanguy pour le temps qu'ils nous ont consacré, le matériel de valeur qu'ils ont mis à notre disposition, le sujet qu'ils nous ont proposé, mais en particulier nous les remercions pour la pédagogie et la sympathie dont ils ont fait preuve à notre égard tout au long de l'année.

Nous remercions également Mme Pascale Gautron pour ses lumières quant aux méthodes de gestion de projet.

Table des matières

Résumé.....	2
Abstract	2
Remerciements	3
Introduction.....	5
Principes de base du radar.....	6
I – Le projet ObsHyper	7
I.1 – Le radar : éléments de base	7
I.2 Montage et acquisitions :.....	10
I.3 – Traitement du signal.....	14
I.4 – Détection	19
I.4.1 Taux de fausses alarmes.....	19
I.4.2 Taux de fausses alarmes constant (CFAR).....	20
I.4.3 Test de l'algorithme CFAR	21
I.4.4 Mise en forme des données	23
I.5 – Principe du tracking :.....	24
I.5.1 – Algorithme des plus proches voisins (K-NN pour K Nearest Neighbours) :.....	24
I.5.2 – Probabilistic data association filter PDAF.....	25
I.5.3 – Multiple Hypothesis Tracking (MHT).....	25
I.5.4 – Filtrage de Kalman	26
I.5.5 - Implémentation.....	27
II – Les principes de l'ingénierie système :	33
II.1 – Exigences et architecture fonctionnelle	33
II.2 – Architecture physique	34
III – L'application de la méthode Agile au sein de l'équipe du projet ObsHyper :.....	38
Conclusion	49
Bibliographie :	50
Table des figures :	50
Table des tableaux :.....	52
Annexes :	52

Introduction

Au cours de ce projet nous avons comme objectif principal de créer en autonomie la totalité de la chaîne composant un radar hyperfréquence afin de pouvoir, à terme, à l'aide de ce dernier, détecter un objet ainsi que sa distance. L'objectif est simple et limité à la détection d'objet à une faible distance. Pour ce faire, l'équipe encadrante et l'ENSTA Bretagne mettent à notre disposition tous les composants nécessaires : générateurs, oscilloscopes, analyseurs, convertisseurs.

Face au re-confinement et à l'éloignement de l'équipe, le projet a été réorganisé pour comporter deux aspects :

- Un aspect pratique avec utilisation du matériel pour répondre à la problématique originelle, qui correspond aux travaux effectués pré-confinement et qui restent pour lors inachevés. Parmi eux et en tête de liste, la prise en main des différents organes à notre disposition, notamment au travers de tests unitaires visant à comprendre le fonctionnement, le rôle, et l'intérêt de chacun d'entre eux, dans le but de pouvoir suggérer une chaîne d'assemblage cohérente et sensée. Cette phase avait lieu sur les créneaux de projet, en lieu et place du bâtiment E et se serait par la suite poursuivi par des essais en extérieur (tout le matériel étant installé sur une baie mobile).
- Un aspect théorique, basé principalement sur le traitement du signal que l'on a acquis. Cet aspect du projet aurait été abordé qu'importe le confinement, toutefois, ce dernier a accéléré les travaux de ce point de vue ci, suivant également les indications de nos encadrants qui nous ont aiguillé sur les différentes parties théoriques que nous abordons plus loin dans ce rapport. On notera au passage que les connaissances acquises grâce au projet nous auront servi de mise en bouche pour les cours qui abordaient ces disciplines lors du 4^e semestre.

Par souci de lisibilité, nous avons pris le parti de grouper l'ensemble des références utilisées (littérature, images, tableaux, etc...) en fin de rapport. Le lien pourra être fait directement par les numéros de référence.

Principes de base du radar

Avant de commencer à rentrer dans les détails de notre projet, il est de bon ton de revenir sur les bases d'un système RADAR. Le principe du RADAR est de détecter une cible mouvante et d'être capable de la suivre. Ceci se fait grâce au phénomène de réflexion électromagnétique : le RADAR envoie une onde avec des caractéristiques bien spécifiques (selon les besoins), et reçoit quelques instants plus tard la réflexion de cette même onde sur la cible en mouvement. Comme bien des phénomènes physiques, cette réflexion est soumise à des pertes qui rendent le signal retour plus ou moins lisible. Celles-ci permettent de caractériser l'efficacité de l'appareil à travers la relation suivante, connue sous le nom de « l'équation du RADAR » :

$$P_r = P_t \frac{G_r G_t \lambda^2 \sigma}{(4\pi)^3 R_r^2 R_t^2}$$

Où P_r et P_t désignent les puissances reçues et transmises (en Watt), G_r et G_t les gains des antennes émettrices et réceptrices (sans dimensions), λ la longueur d'onde exploitée par le RADAR (en mètres), σ la surface équivalente RADAR qui équivaut au coefficient de réflexion de la cible (en mètres carrés, nous reviendrons dessus), et enfin R_r et R_t les distances cible-émetteur et cible-récepteur (en mètres).

Une autre grandeur importante est la portée maximale du radar. Si l'on considère que l'antenne émettrice est l'antenne réceptrice, ce qui sera notre cas, on peut confondre G_r et G_t et noter ces deux grandeurs G , la portée maximale se définit ainsi par la relation :

$$R_{max}^4 = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 S_{min}}$$

Avec S_{min} représentant le signal minimum que l'on peut détecter à travers la puissance reçue (en W/m^2).

Nous devons également introduire la notion de résolution en distance comme l'aptitude d'un RADAR à distinguer deux cibles dans la même direction, mais à des distances différentes. Celle-ci dépend fondamentalement de la largeur temporelle de l'impulsion émise (notée τ) et se définit par la relation :

$$S_r = \frac{c_0 \tau}{2}$$

Un autre phénomène auquel nous allons être confrontés est celui de l'ambiguïté. Celle-ci intervient si l'écho d'une impulsion émise revient au radar après qu'une seconde impulsion ait été émise. Dans ce cas il est impossible de distinguer si l'écho reçu est la réflexion de la première émission ou de la seconde. On définit donc une portée maximale sans ambiguïté par la relation :

$$R_{max} = \frac{c_0(T - \tau)}{2}$$

Où T désigne la période d'émission des impulsions et τ la durée d'une impulsion (en secondes).

Enfin, parlons de la surface équivalente RADAR (notée SER). Elle permet de quantifier la partie de l'énergie qui est réfléchie par une cible en direction du radar en comparaison à l'énergie incidente. Elle se formalise par l'expression :

$$\sigma = \frac{4\pi r^2 S_r}{S_t}$$

Avec S_t désignant l'énergie rétrodiffusée par la cible et S_r , l'énergie reçue par cette même cible à une distance r (énergies en W/m^2).

Pour finir, dans le cadre de notre projet nous prenons le parti de ne pas nous attarder sur l'effet Doppler-Fizeau, puisque nous n'allons pas réellement suivre une cible en mouvement, comme vous pourrez le lire plus bas, mais plutôt simuler un suivi de mouvement.

I – Le projet ObsHyper

I.1 – Le radar : éléments de base

Un radar hyperfréquence est en général constitué de 5 éléments : un émetteur, un duplexeur, une ou plusieurs antennes, un récepteur et enfin un moyen d'affichage. L'affichage le plus classique étant le Plan Position Indicator (PPI) qui restitue une vue en deux dimensions de l'espace à 360 degrés autour du radar.

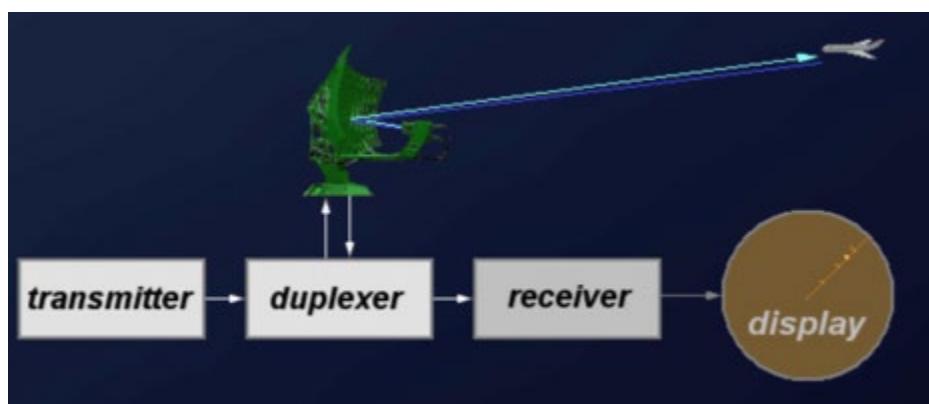


Figure 1 : Chaîne radar type

Dans notre cas, le duplexeur n'est pas nécessaire car le système possède 2 antennes, une pour l'émetteur et une pour le récepteur. Ainsi le duplexeur qui est un commutateur qui relie alternativement l'antenne à l'émetteur puis au récepteur radio n'est pas nécessaire.

Pour notre projet, l'émetteur (ou générateur de signaux) sera le générateur de signaux en bande de base RS AFQ 100B. Dans le jargon des télécommunications, le terme de bande de base (ou en anglais baseband) désigne une technique de transmission dans laquelle le signal est envoyé directement sur le canal après Codage en ligne sans passer par un codage canal (sans

modulation). Ce générateur peut produire des signaux radar complexes avec des impulsions courtes et des temps de montée et de descente rapide. Il possède également un bon rapport signal sur bruit. Les impulsions émises sont de très courtes durées, ce qui permet à la bande passante d'atteindre de très grandes valeurs, on parle alors d'UltraLargeBand (ULB), utilisée notamment pour la communication en haut débit sur courte distance.

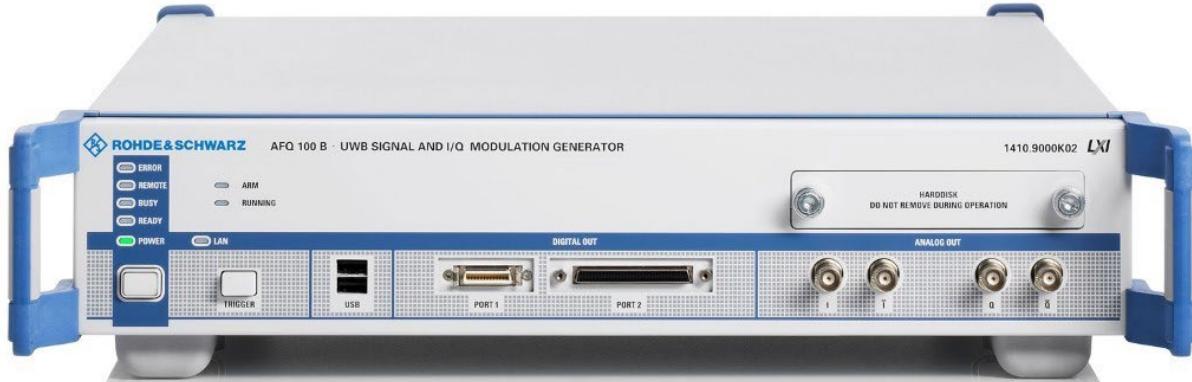


Figure 2 : RS AFQ 100B

Grâce au logiciel K6 Pulse Sequencer de Rohde & Schwarz, nous avons pu créer un signal complétement personnalisé. En partant d'un chirp de 500MHz, nous avons configuré notre signal tel que décrit par l'image infra. En effet, pour éviter l'ambiguïté (détailée plus tard), il était nécessaire de mettre un "off-time" conséquent relativement à la durée du signal.

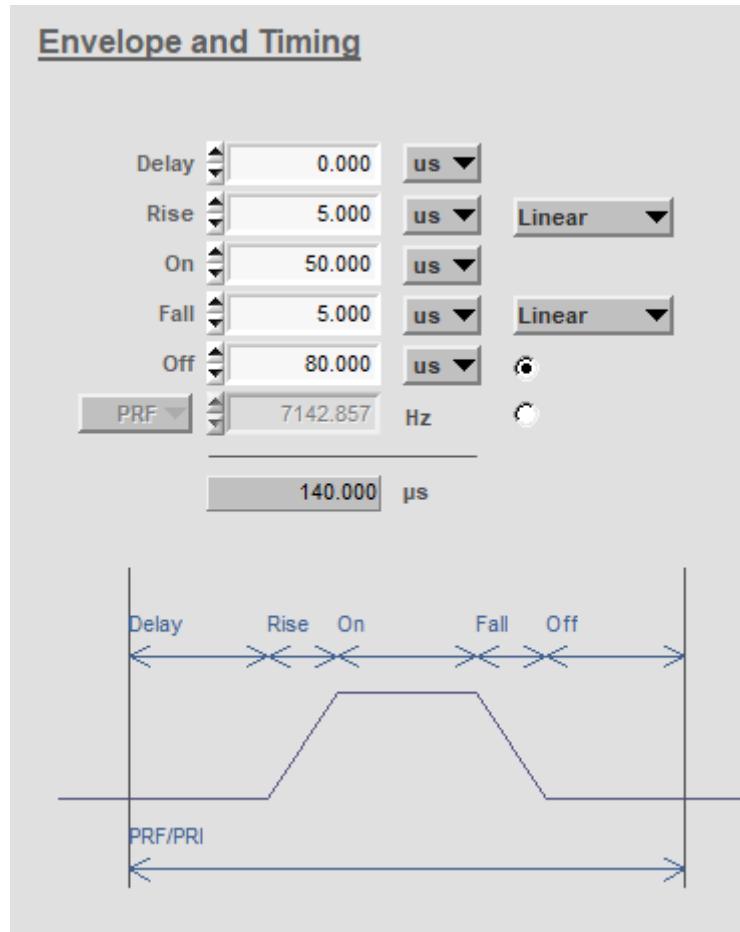


Figure 3 : Configuration d'un chirp spécifique sur K6 Pulse Sequencer

Le générateur de signaux en bande de base RS AFQ 100B est ensuite couplé au générateur de signaux vectoriels RS SMBV 100 A qui transpose les signaux en bande de base en signal RF. Ce dernier pourrait servir de générateur signaux mais dispose de caractéristiques moins intéressantes que son homologue. Il sert uniquement dans notre cas de modulateur numérique et analogique : après avoir généré le signal il le transforme pour l'adapter au canal de transmission. Ici le canal est hertzien sachant qu'il aurait pu être filaire ou optique. C'est la modulation. On va donner à l'onde dite porteuse de nouvelles caractéristiques d'amplitude de fréquences et de phases. On a vu que ce générateur avait la particularité de posséder une modulation de type IQ, La modulation d'amplitude en quadrature qui modifie la porteuse elle-même et une onde en quadrature autrement dit déphasé de 90 degrés. Ce qui permet de modifier simultanément la phase et l'amplitude.



Figure 4 : RS SMBV 100A

Enfin, le générateur de signaux vectoriels RS SMBV 100A est branché au convertisseur GeoSync Microwave. Ce dernier sert à la fois de Up Converter à l'émission et de Down Converter à la réception. En d'autres termes, son rôle est d'effectuer une translation en fréquence entre la bande L (fournie et reçue par le générateur de signaux vectoriels) et des fréquences décryptables pour le transpondeur qui reçoit l'onde électromagnétique émise par l'antenne.



These block converter systems provide frequency translation between the transponder band and L-band frequencies.

The two independent converter assemblies are “hot swappable” through the rear of the chassis. Each converter tray has independent control of the front panel. The front panel can be replaced without loss of signal.

BLOCK DOWNCONVERTERS

Input (GHz)	Output (MHz)	LO (GHz)	Conv Tray Model #
9.75-10.25	250-750	9.5	D10
14.0-14.5	250-750	13.75	D14.25
14.75-15.25	250-750	14.5	D15

BLOCK UPCONVERTERS

Input (GHz)	Output (GHz)	LO (GHz)	Conv Tray Model #
250-750	9.75-10.25	9.5	U10
250-750	14.0-14.5	13.75	U14.25
250-750	14.75-15.25	14.5	U15

Figure 5 : Convertisseur GeoSync Microwave

Finalement, il est éventuellement possible d'analyser le signal avec un oscilloscope. Le schéma complet est donc le suivant :

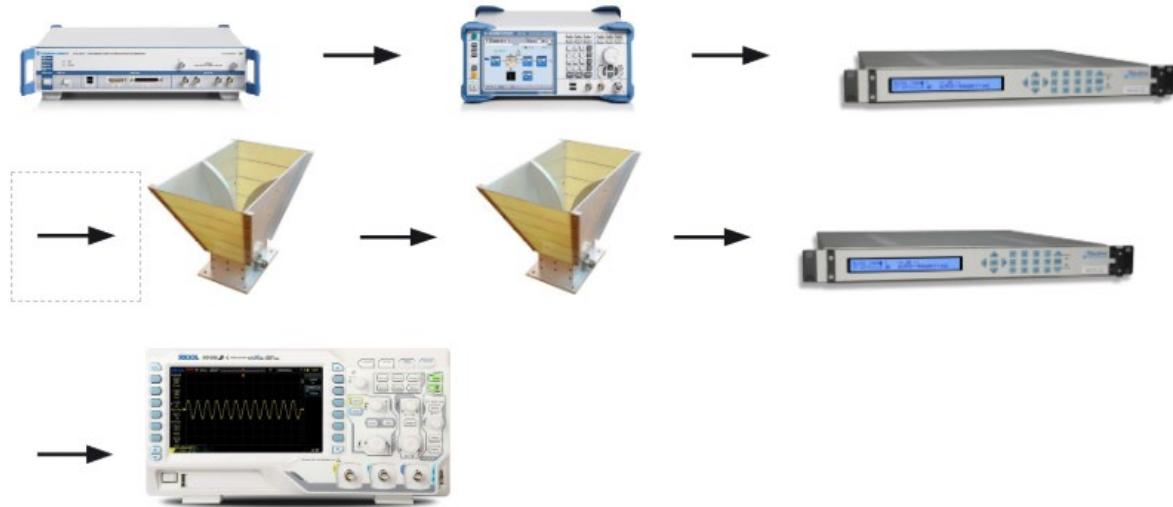


Figure 6 : Montage type de notre système

I.2 Montage et acquisitions :

Comme avancé dans le rapport de mi-parcours^[1], le montage est par la suite devenu celui-ci :

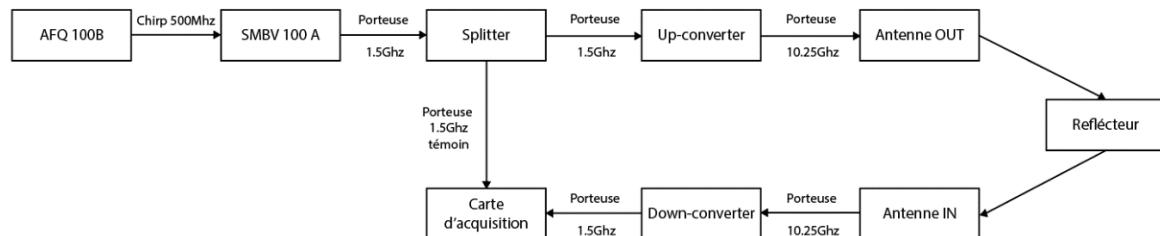


Figure 7 : Chaîne fonctionnelle de notre système

La présence du splitter permet de procéder à une acquisition sur deux canaux, d'une part le signal en sortie de l'antenne réceptrice et d'autre part le signal tel qu'il est émis. L'avantage est de pouvoir procéder à la corrélation mathématique de ces deux signaux et ainsi d'effectuer, en post-traitement, un filtrage dit "adapté". Le principal inconvénient d'un tel montage réside dans l'atténuation causée par une multiplication des canaux, par instants. Il semblerait même qu'on puisse perdre tout signal retour au sein d'une pièce fermée, comme nous avons pu l'expérimenter.

Toutefois, après des tests semi-unitaires (puisque chaque test sur un appareil, nécessite le concours des précédents), nous avons obtenus des résultats relativement probants pour une telle chaîne qui aboutissent, sans traitement, aux analyses spectrales suivantes :

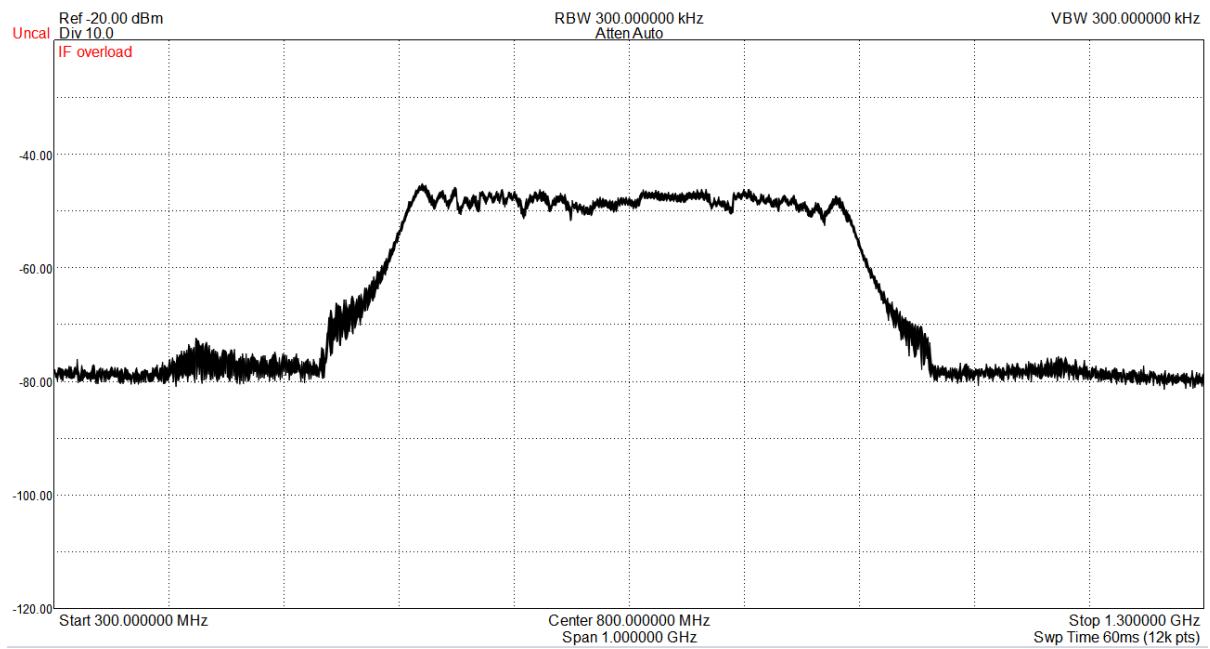


Figure 8 : Signal témoin en sortie du splitter

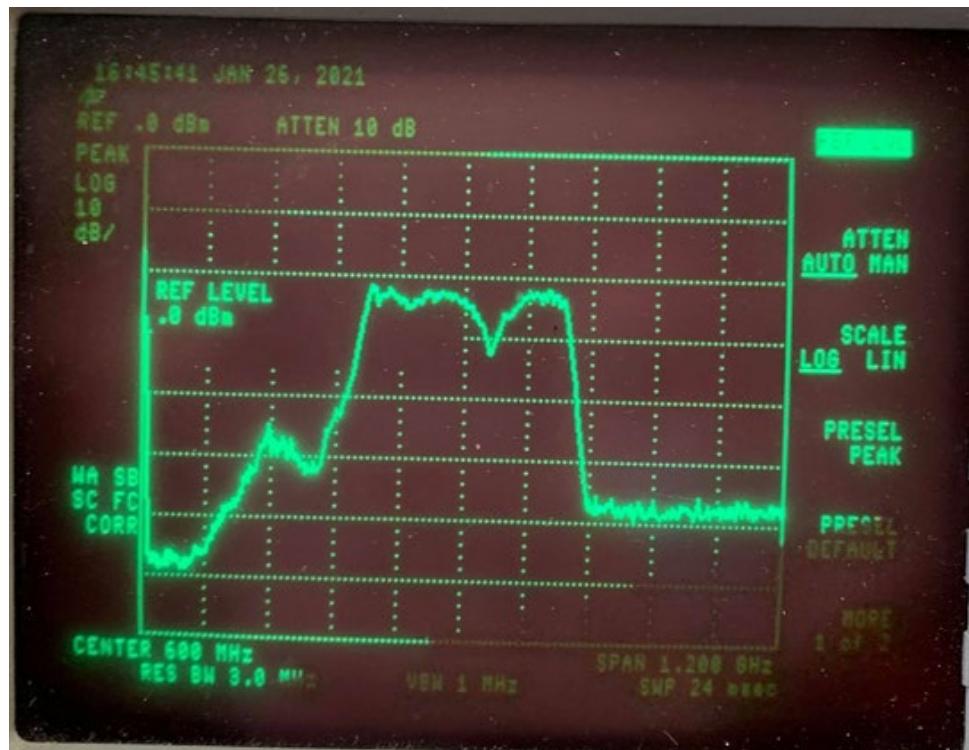


Figure 9 : Signal lu après réflexion sur un mur à 1 mètre

Nous adoptons donc cette chaîne, que nous mettons en œuvre dans le montage suivant qui permet de simuler un suivi de cible en mouvement.



Figure 10 : Montage réel du système ObsHyper (gauche) – cible (droite)

Afin de donner du sens à nos algorithmes, il nous faut une cible mobile. Seulement, les durées des acquisitions que nous pouvions effectuer étaient de l'ordre de la milliseconde. On comprend alors bien la difficulté de faire bouger une cible sur une durée si courte. Pour contourner le problème, nous allons exploiter l'une des particularités du matériel à notre disposition : toutes les impulsions sont émises avec la même phase initiale, ce qui permet de simuler le déplacement d'une cible. En effet, si le déphasage entre deux séries d'impulsions est inférieur à 2π (i.e. environ 3 centimètres), on peut lire un déplacement sans prendre le risque de confondre le signal avec un écho. Pour y parvenir, nous avons disposé au sol un mètre permettant un déplacement relatif suffisamment précis.

Pour la réflexion du signal, nous avions à disposition plusieurs réflecteurs (plan comme sur la photo, sphérique avec un axe de réflexion préférentiel, ou conique), mais finalement nous avons choisi le plus simple d'entre eux, une surface métallique plane. L'amplitude lue est la meilleure parmi tous, bien que son maniement soit assez exigeant, en particulier en termes de positionnement angulaire. Il faut bien que la plaque soit orthogonale aux antennes.



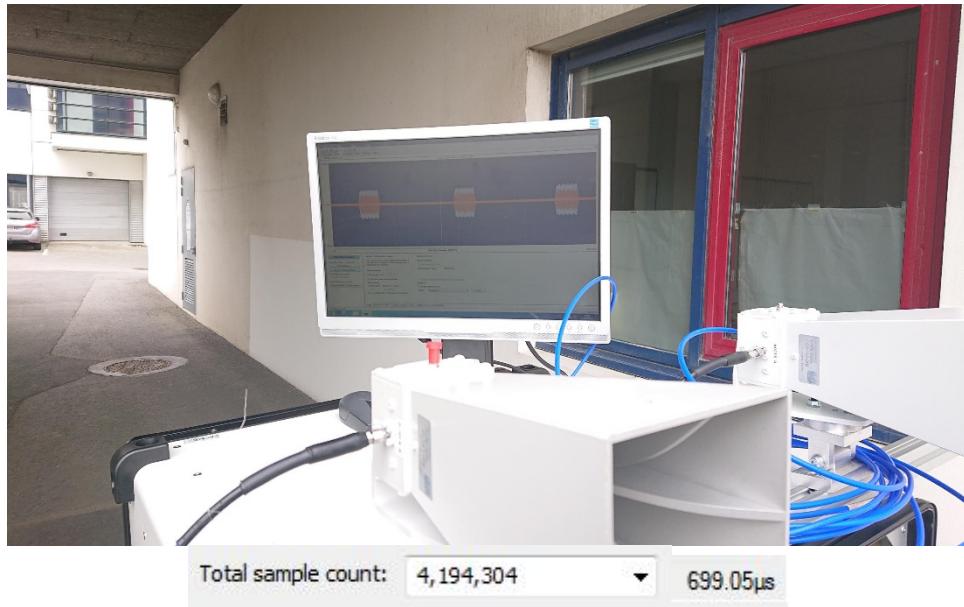
Figures 11 et 12 : Positionnement angulaire correct du réflecteur (gauche) – amplitude réfléchie (droite)



Figures 13 et 14 : Positionnement angulaire incorrect du réflecteur (gauche) – amplitude réfléchie (droite)

Sur ces figures, on lit en blanc le signal en sortie du splitter et en orange le signal en sortie du down-converter et donc après réflexion. On peut constater qu'un décalage angulaire de l'ordre d'une dizaine de degrés suffit à perdre toute trace de la cible, le bruit orange restant sur la figure 14 étant uniquement du bruit de fond, présent avec ou sans cible.

On effectue des acquisitions successives de 4 194 304 points (pour une durée de 699.05us), ce qui représente environ six émissions, espacées physiquement d'environ 2 centimètres chacune.



Figures 15 à 17 : Acquisition continue, nombre de points et durée de l'acquisition

Nous avons effectué 42 acquisitions de cette façon de 1m97 à 2m29, puis de 5m à 5m28. Celles-ci sont récupérées et exploitées sous forme de fichiers texte, comprenant deux colonnes (pour les deux signaux) de nombres de 0 à 255 (127 correspondant au 0 volt de la carte d'acquisition). Ceux-ci ont été bases sur lesquelles ont travaillé nos algorithmes. Les résultats sont disponibles à la fin du chapitre I.5.5, seulement, afin d'en comprendre l'origine, il est suggéré de suivre les chapitres de ce rapport dans l'ordre et de se laisser guider par le raisonnement que nous avons suivi.

I.3 – Traitement du signal

Pour analyser un signal reçu, il est intéressant de l'afficher en tant que tel et d'afficher son spectre. Python permet cette analyse via les fonctions de l'algorithme 1, fourni en annexe :

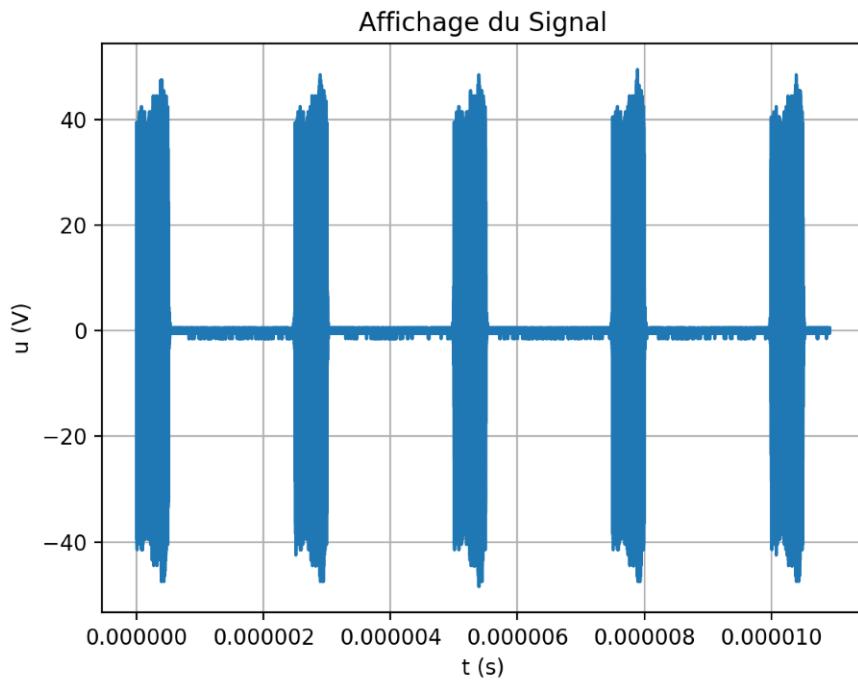


Figure 18 : Signal obtenu en sortie d'algorithme

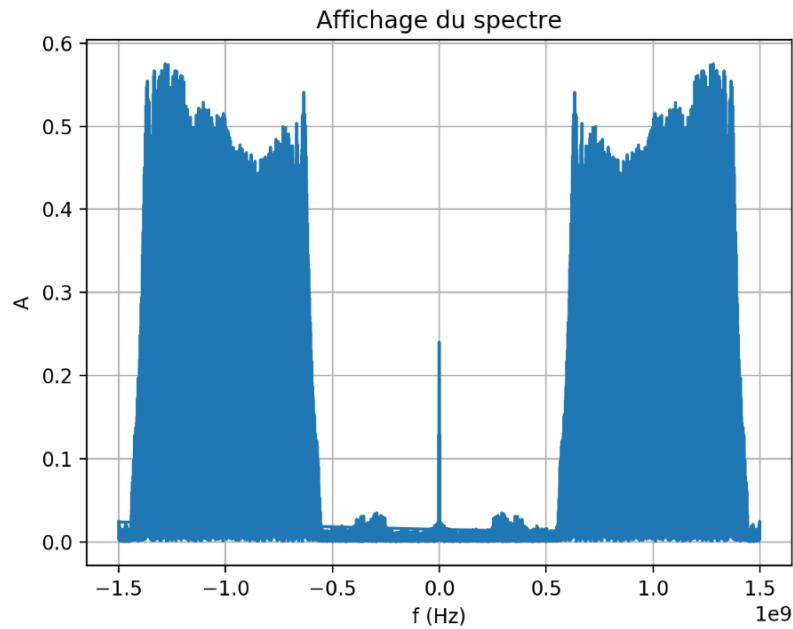


Figure 19 : Spectre obtenu en sortie d'algorithme

On remarque que la largeur de bande est très importante, cela est dû au fait que les impulsions émises sont des chirps (mot d'origine anglaise signifiant "gazouillis"). Ils sont par définition des signaux pseudopériodiques modulés en fréquence autour d'une fréquence porteuse. Le chirp linéaire, le plus courant d'entre eux, correspond à un signal à rampe de fréquence linéaire de la forme $c(t) = Ae^{i(a.t+b).t+c}$.^[2]

Dans les applications radar et sonar, le chirp linéaire est souvent le signal utilisé pour réaliser la compression d'impulsion. L'impulsion étant de durée finie, le spectre d'amplitude est une fonction porte. Le signal est de durée T , débute à $t = 0$ et balaie la bande de fréquence Δf centrée sur f_0 .

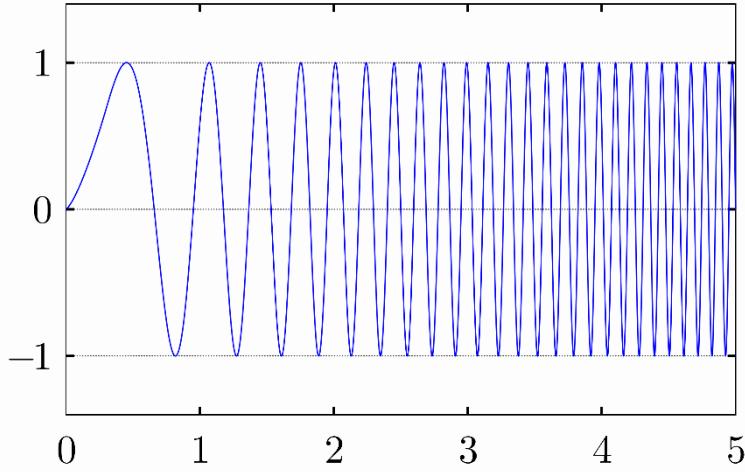


Figure 20 : Exemple de chirp linéaire

La compression d'impulsion (en anglais, *pulse compression*) est une technique de traitement du signal utilisée afin d'augmenter la résolution en distance de la mesure ainsi que le rapport signal sur bruit, par modulation du signal émis. L'idée générale est d'obtenir une impulsion longue, afin de conserver une énergie suffisante à la réception, sans pour autant sacrifier la résolution par rapport à une impulsion courte de puissance équivalente. Pour cela, on génère un signal dont le support temporel est relativement long (pour maximiser l'énergie émise). Cependant, on module ce signal de telle manière qu'après un filtrage adapté, on puisse différencier les signaux de retours de plusieurs cibles qui pourraient se chevaucher à l'intérieur de la distance que représente la longueur de l'impulsion. En effet, comme chaque partie de l'impulsion a sa propre fréquence, les retours sont complètement séparés.^[3]

Par exemple, si on génère le signal le plus simple que peut émettre un radar à impulsions : un train de signaux sinusoïdaux, d'amplitude A et de fréquence f_0 , tronqué par une fonction porte de longueur T :

$$s(t) = \begin{cases} Ae^{i2\pi f_0 t} & \text{si } 0 \leq t \leq T \\ 0 & \text{sinon} \end{cases}$$

Le signal réfléchi est alors de la forme :

$$r(t) = \begin{cases} K * Ae^{i2\pi f_0(t-t_r)} + B(t) & \text{si } t_r \leq t \leq t_r + T \\ B(t) & \text{sinon} \end{cases}$$

Où $B(t)$ est un bruit blanc gaussien (ce qui est généralement le cas dans la réalité). Un bruit blanc est une réalisation d'un processus aléatoire dans lequel la densité spectrale de puissance est la même pour toutes les fréquences de la bande passante. Le bruit blanc gaussien est un bruit blanc qui suit une loi normale de moyenne et variance données. K est un facteur d'atténuation inférieur à 1 qui correspond à la perte de puissance dans le canal de transmission et t_r le temps que met le signal pour revenir au détecteur. L'intercorrélation de ces deux signaux (dans le cadre du filtrage adapté) donne

$$\langle s, r \rangle (t) = K \cdot A^2 \cdot \Lambda \left(\frac{t - t_r}{T} \right) e^{i2\pi f_0(t-t_r)} + B'(t)$$

Avec $\Lambda(t) = \begin{cases} 1 + 2t & \text{si } -1/2 < t \leq 0 \\ 1 - 2t & \text{si } 0 < t \leq 1/2 \\ 0 & \text{sinon} \end{cases}$ et $B'(t)$ un bruit blanc gaussien.

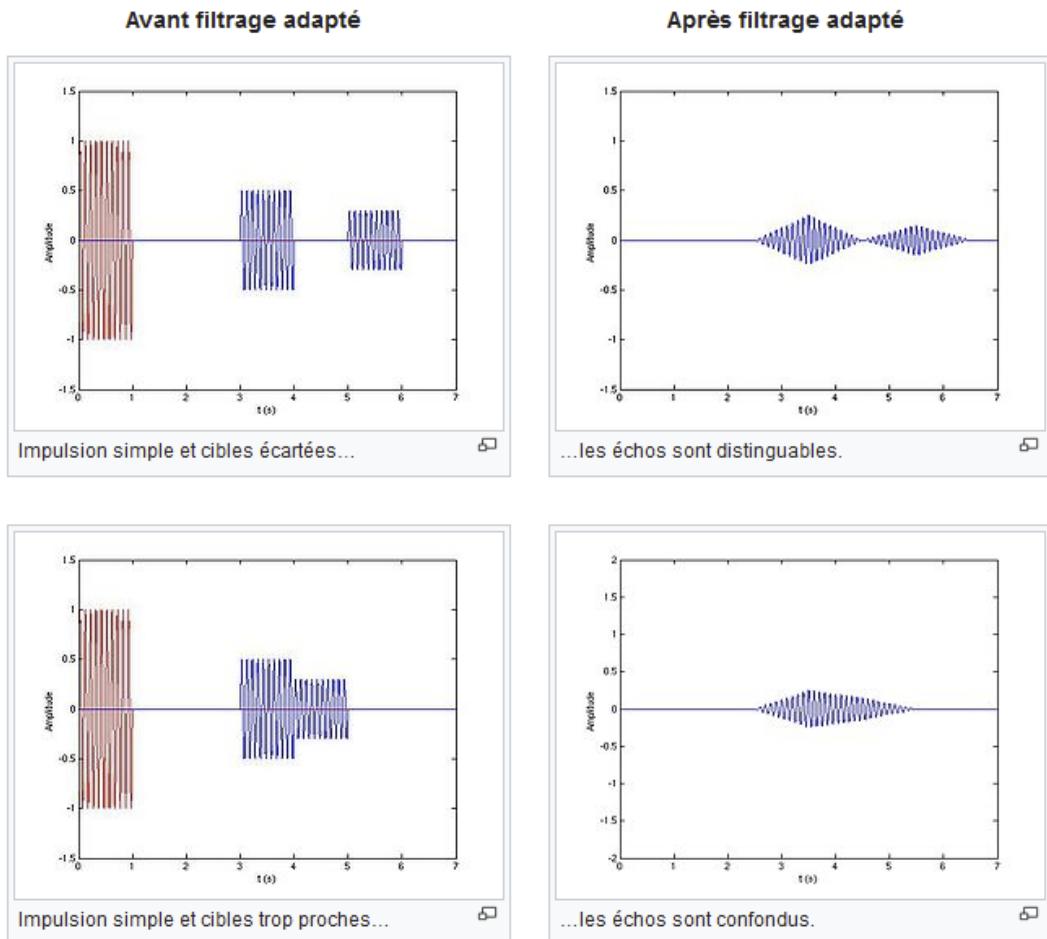


Figure 21 : Signal émis en rouge et échos atténuerés en bleu

On remarque que la largeur des impulsions échos après filtrage adapté est à peu près équivalente à celle de ces mêmes signaux avant l'intercorrélation avec le signal émis. On voit donc que les signaux échos après filtrage sont confondu dès que les signaux échos avant filtrage sont joints. Par conséquent, si les impulsions sont de durées T et se déplacent à la vitesse c , la résolution

spatiale est $cT/2$. La division par deux correspond à la réalité de retour de l'onde après écho sur la cible.

Cette fois on utilise un chirp (compression d'impulsion) à l'émission :

$$s(t) = \begin{cases} Ae^{i2\pi(f_0 + \frac{\Delta f}{2T}t)t} & \text{si } -T/2 \leq t \leq T/2 \\ 0 & \text{sinon} \end{cases}$$

L'intercorrélation est alors de la forme ci-après :

$$\langle s_c, s_c \rangle(t) = T \cdot \Lambda\left(\frac{t}{T}\right) \cdot \text{sinc}\left(\pi\Delta ft \cdot \Lambda\left(\frac{t}{T}\right)\right) \cdot e^{i2\pi f_0 t}$$

C'est la présence du sinus cardinal dans l'intercorrélation qui réduit la largeur de l'impulsion filtré. La largeur temporelle de ce sinus cardinal à -3 dB est plus ou moins égale à $T' \approx 1/\Delta f$. Tout se passe donc comme si après compression d'impulsion, on avait la résolution d'une impulsion simple de durée T' qui, pour les choix courants de Δf , est plus petite que T , ce qui justifie le nom de compression d'impulsion. Le résultat est alors le suivant :

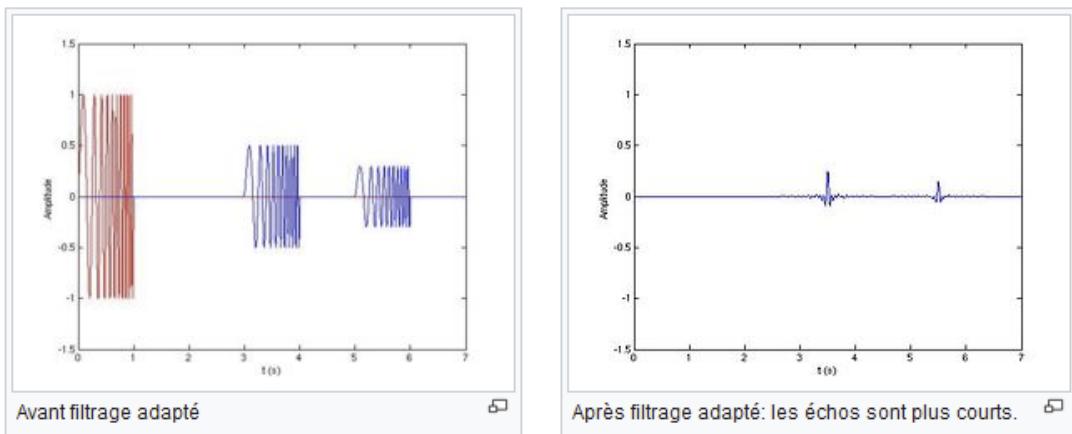


Figure 22 : Signal émis en rouge et échos atténusés en bleu

On constate bien une réduction de la largeur des signaux échos filtré, ce qui induit une augmentation de la résolution spatiale.^[4]

Passons alors à la mise en place du filtrage adapté : c'est un filtrage linéaire qui permet d'optimiser le rapport signal sur bruit (SNR). Il est utilisé pour la détection d'un signal inconnu en corrélation avec un signal connu. D'un point de vue technique, cela revient à effectuer une intercorrélation entre le signal émis, connu, et le signal reçu, inconnu et bruité. Si on considère une seule impulsion, le résultat est un sinus cardinal centré sur t_{delay} , le temps de retard du signal reçu par rapport au signal émis.

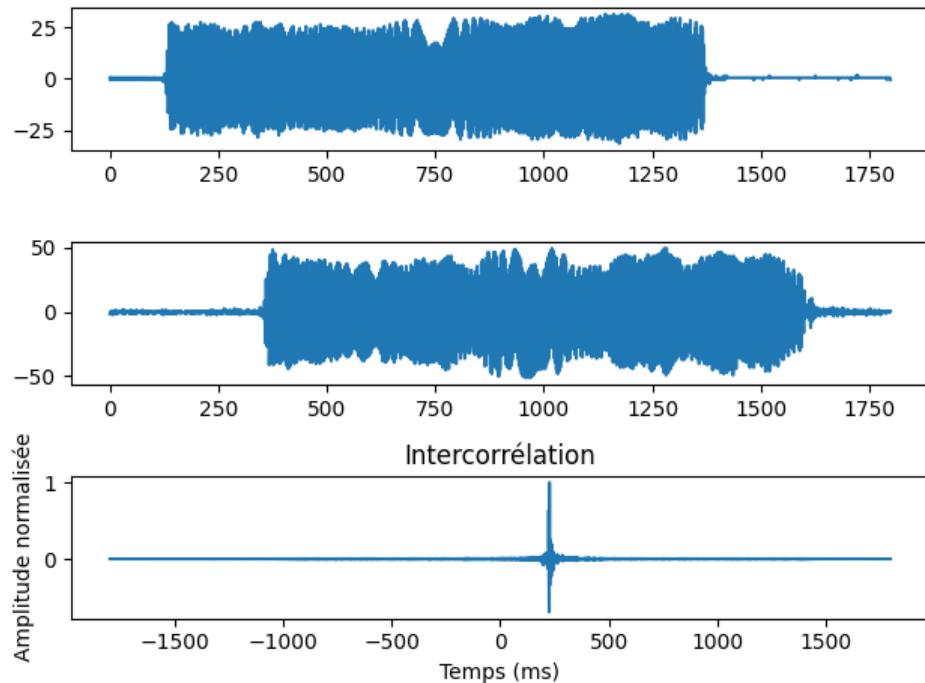


Figure 23 : Exemple d'intercorrélation entre un signal émis et son retour différé et bruité

Lorsqu'on effectue l'intercorrélation sur l'intégralité des signaux reçus et émis (l'entièreté du train d'impulsion), le résultat est un ensemble de $2n - 1$ sinus cardinaux, avec n le nombre d'impulsions.

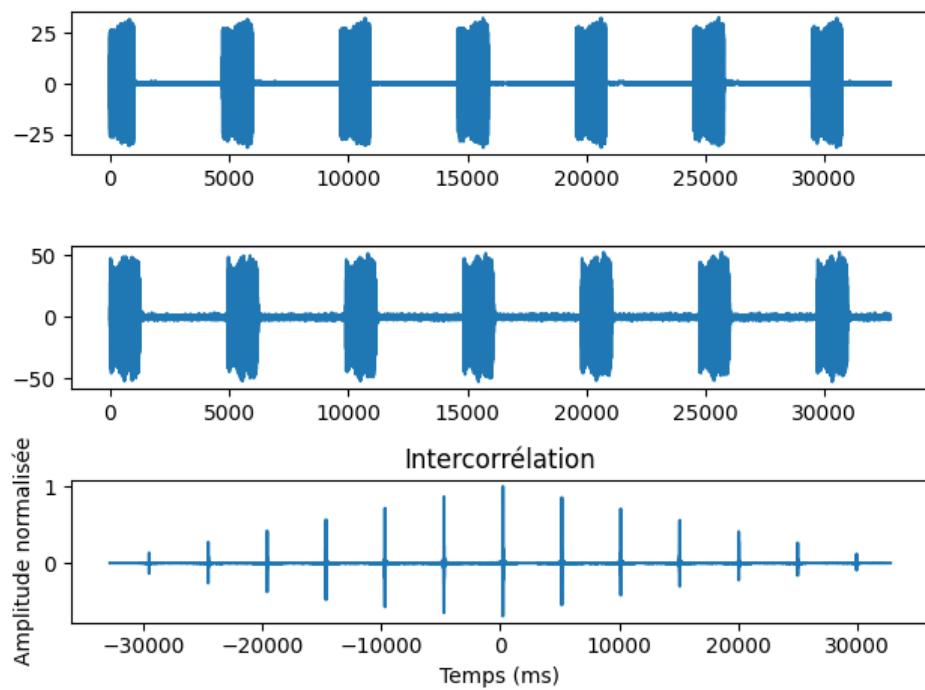


Figure 24 : Exemple d'intercorrélation entre une succession de signaux émis et leur retours différés et bruités

Cela s'explique par le fait que l'intercorrélation est une mesure de similitude entre deux signaux par translation temporelle. Ainsi, le sinus cardinal central est le sinus cardinal avec l'amplitude la plus forte car il correspond au recouvrement exact des deux trains d'impulsions. Le

sinus cardinal à sa droite (respectivement à sa gauche) correspond à un recouvrement exact moins la première impulsion reçue et la dernière impulsion émise (respectivement la première impulsion émise et la dernière impulsion reçue). Ainsi l'amplitude de ces sinus cardinaux est moindre. Pour les sinus cardinaux suivants, on continue la translation et on répète le même raisonnement, jusqu'à arriver aux sinus cardinaux extrêmes. Ces sinus cardinaux témoignent du recouvrement des impulsions extrêmes des signaux émis et reçus avec perte de toutes les autres impulsions du train. Perte due à la translation temporelle de l'intercorrélation.

Afin de vérifier expérimentalement la cohérence de ce modèle, bien qu'elle ne soit plus à prouver, nous avons réalisé les tests unitaires aboutissant aux deux figures ci-dessus (23 et 24). Il s'agit ici, après avoir validé notre montage de la chaîne physique, d'exploiter une impulsion unique, en espérant retrouver un sinus cardinal, puis plusieurs impulsions (avec antenne et cibles fixes) en espérant cette fois retrouver une forme de sinus cardinaux d'amplitudes croissantes puis décroissantes, régulièrement espacés. Comme vous pouvez le constater, nos attentes théoriques ont été honorées, ce qui confirme sans surprises la pertinence du filtrage adapté.

I.4 – Détection

La détection consiste à mettre en évidence les cibles potentielles, à partir du signal reçu et préalablement traité. Pour ce faire il faut être en mesure d'isoler les pics correspondants à des cibles du bruit environnant. La complexité de la manipulation est de réussir à faire ceci en évitant de provoquer de fausses alarmes dues au bruit. Pour ce faire nous utilisons un seuil de détection, lorsque ce seuil est dépassé nous considérons qu'il y a présence d'une cible. Néanmoins puisque qu'un bruit ou d'autres signaux peuvent interférer et eux aussi dépasser ce seuil, nous sommes confrontés à de fausses alarmes. Il faut donc une façon robuste de choisir le seuil de détection afin de limiter le taux de fausses alarmes, c'est ce que nous allons développer ici.

I.4.1 Taux de fausses alarmes

La première notion importante du problème de détection, que nous souhaitons maîtriser au mieux est donc le taux de fausses alarmes, c'est un facteur primordial pour assurer la fiabilité de notre système radar.

Une fausse alarme est « une décision de détection de cible radar erronée causée par du bruit ou d'autres signaux interférents dépassant le seuil de détection »^[5]. Le taux de fausses alarmes (FAR) est calculé à l'aide de la formule suivante :

$$FAR = \frac{\text{Fausses cibles par PRT}}{\text{Nombre de cellules}}$$

Ainsi de fausses alarmes sont générées lorsque le bruit dépasse un niveau seuil prédéfini, par la présence de signaux parasites.

- Si le seuil est réglé trop haut : il y aura très peu de fausses alarmes mais le rapport signal sur bruit requis empêchera la détection de cibles valides.
- Si le seuil est réglé trop bas : le grand nombre de fausses alarmes masquera la détection de cibles valides.
- Si le seuil est réglé de façon optimale : probabilité de détection = 83%, mais une fausse alarme se produit, $FAR = 1.5 * 10^{-3}$.
- Le seuil est défini de façon variable : taux de fausses alarmes constant.

Le taux de fausses alarmes dépend du niveau de toutes les interférences (bruit, encombrement, brouillage). A courte distance l'influence du clutter fixe (ensemble d'échos retournés par l'environnement) est plus importante que le bruit. A longue distance c'est le plus bruit qui a le plus gros impact. Donc le taux de fausses alarmes dépend de la plage. Il faut noter que le taux de fausses alarmes augmente à courte distance.

I.4.2 Taux de fausses alarmes constant (CFAR)

Afin de maîtriser ce point clé nous allons utiliser un schéma conservant un taux de fausses alarmes constant et faisant varier le seuil de détection en fonction de fonction de l'environnement. Ainsi le seuil de détection restera au-dessus du niveau du bruit ce qui nous permettra d'isoler les cibles.

Il existe un grand nombre de type de circuit CFAR, qui sont pour la plupart basés sur le CFAR à moyenne de cellules que nous allons mettre en œuvre ici.

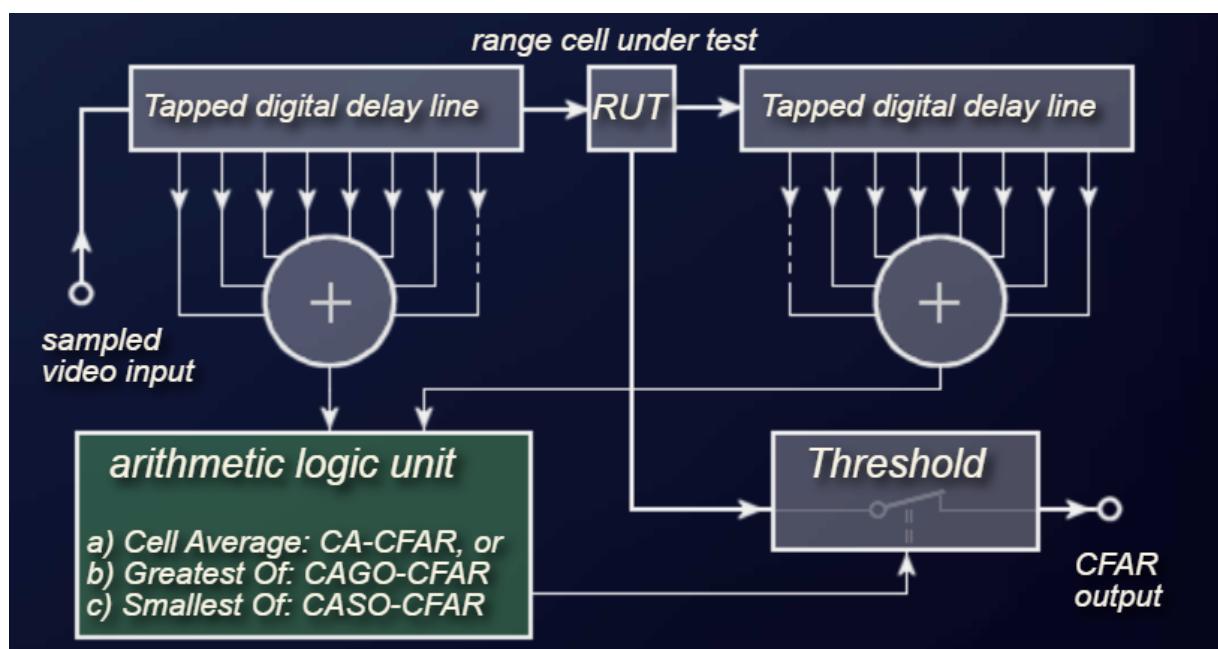


Figure 25 : Principe du "Cell-averaging CFAR"

Le signal reçu est découpé en "cellules" permettant de comparer sa valeur à un instant donné à ses valeurs précédentes et suivante. Ainsi le circuit estime le niveau d'interférence (clutter et bruit) grâce aux cellules en amont et en aval de la cellule sous test, et utilise cette estimation pour décider s'il y a ou non une cible dans la cellule sous test. Le bruit est considéré comme homogène sur la plage testée donc l'estimation faite autour de la cellule de test est valable pour cette dernière. Ce processus est répété sur toutes les cellules. Nous aurons théoriquement un taux de fausses alarmes constant qui est indépendant du niveau du clutter et de bruit. De plus des cellules de gardes sont ajoutées afin de ne pas prendre en compte le signal écho présent juste avant et juste après la cible, qui présente une forte amplitude.^[6]

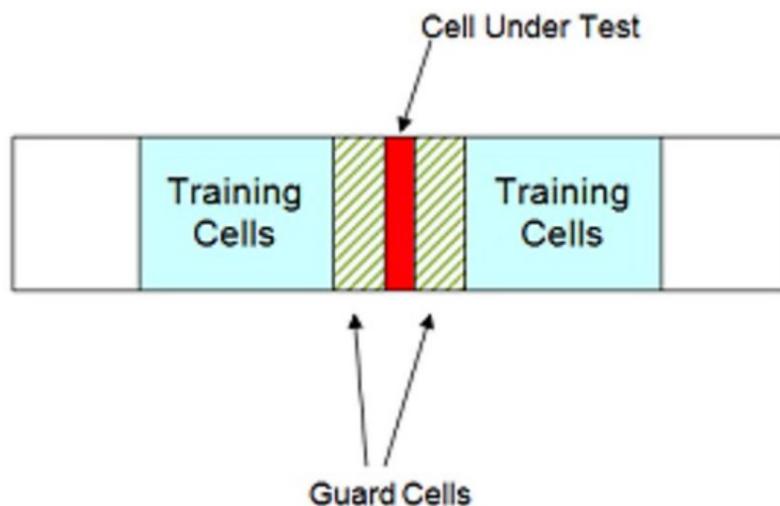


Figure 26 : Schéma de principe des cellules testées

I.4.3 Test de l'algorithme CFAR

Une première implémentation de l'algorithme CFAR a été réalisé en python, présenté en annexe. Il est immédiatement apparu que cet algorithme est très sensible au nombre de cellules d'entraînement et au nombre de cellules de gardes que nous choisissons. Tout d'abord l'algorithme a été testé sur des sinusoïdes bruitées générées de façon aléatoire. Les résultats sont présentés ci-dessous, où les signaux considérés comme cible par l'algorithme sont marqués par un point rouge :

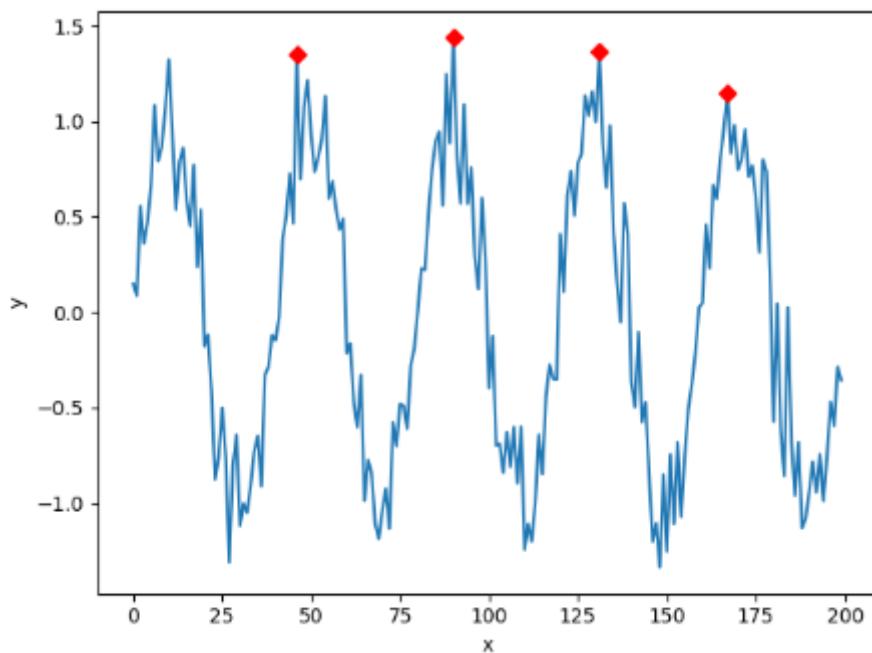


Figure 27 : Résultat de l'algorithme CFAR

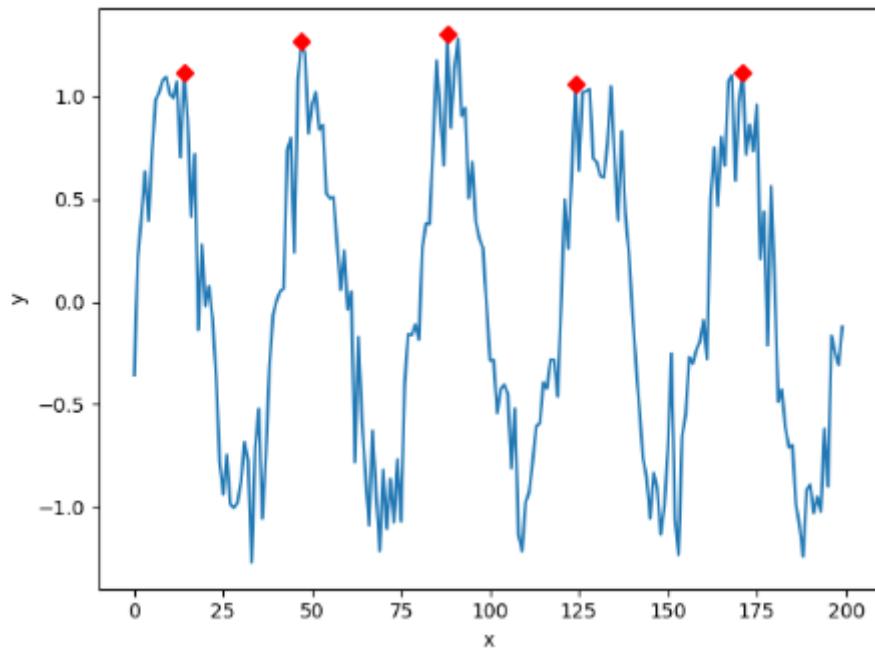


Figure 28 : Résultat de l'algorithme CFAR

Par la suite, l'algorithme a été testé sur le signal acquis avant le re-confinement et présenté à la partie précédente, nous avons obtenus les résultats ci-dessous :

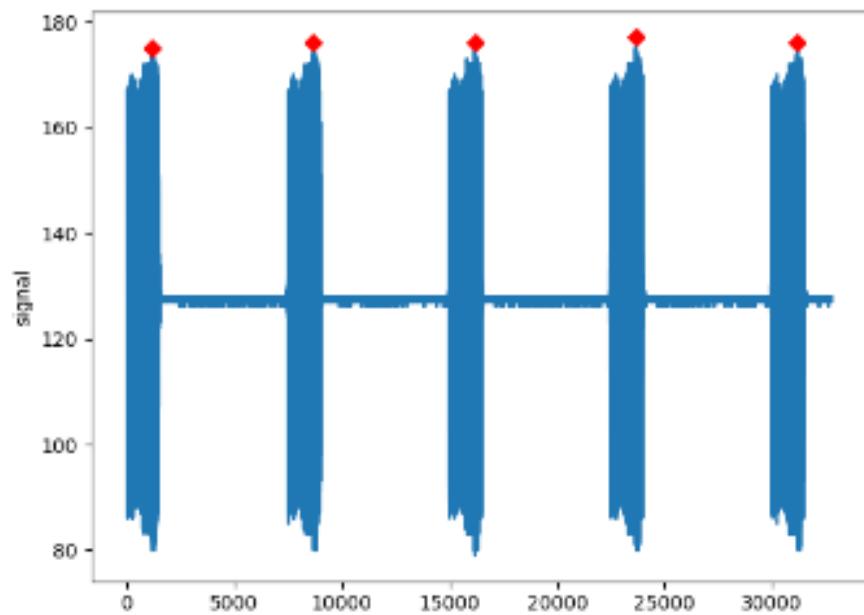


Figure 29 : Résultat de l'algorithme CFAR sur un signal type de nos appareils

Afin d'améliorer le signal sur lequel nous travaillons, un filtrage adapté a été appliqué au signal brut. Afin que ce nouveau signal soit correctement interprété par le CFAR, il a fallu adapter les paramètres d'entrée de l'algorithme, notamment le nombre de cellules d'entraînement et le nombre de cellules de gardes.

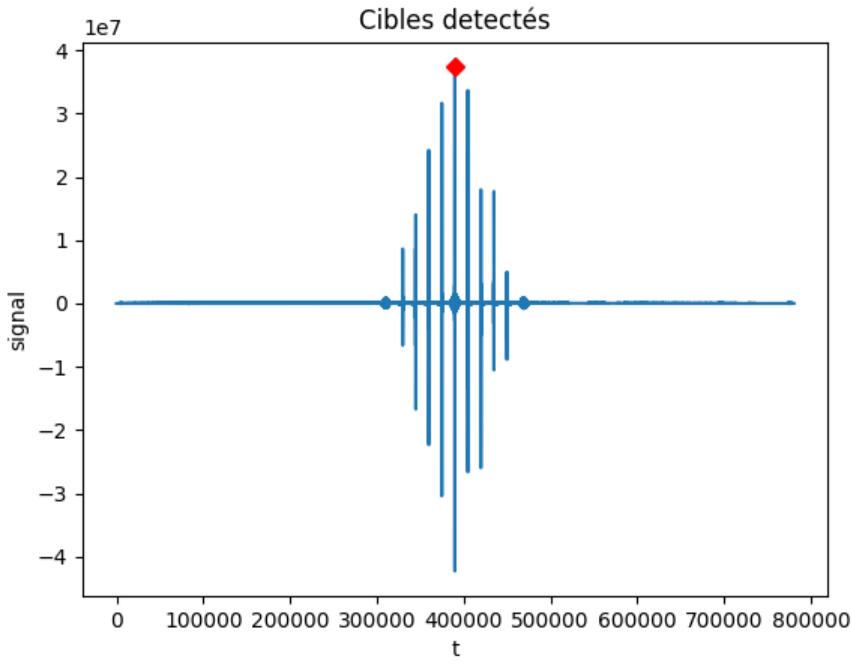


Figure 30 : Détection de cible après filtrage adapté sur un signal brut

Les résultats sont plutôt concluants puis que l'algorithme est en mesure d'isoler le pic correspondant à la cible à chaque fois sans se laisser tromper par la forme du signal qu'il traite.

I.4.4 Mise en forme des données

En parallèle des acquisitions menées, il a été nécessaire de penser une refonte du code afin de rendre la chaîne de post traitement fonctionnelle. En effet, plusieurs raisons ont motivé ceci. Tout d'abord toutes les parties du code ne travaillaient pas avec les mêmes éléments, certaines travaillaient sur des listes alors que d'autres sur des arrays numpy, ceci empêchait une correcte communication des différents éléments. Un second point de blocage fut d'être en mesure de pouvoir associer les signaux retours au chirp qui lui correspond. Pour remédier à ces deux problèmes l'intégralité du code a été modifiée pour ne travailler qu'avec des matrices. Ainsi le signal brut est en premier lieu découpé en morceaux correspondants aux impulsions successives et à leur temps d'écoute associé. Finalement les lignes de la matrice correspondent chacune à une impulsion et son temps d'écoute, donc une ligne contient une période d'émission. Cette configuration nous permet de savoir facilement quel retour est associé à quelle émission. Cette matrice de signal retour brut est ensuite soumise au filtrage adapté sous forme matriciel, chaque ligne subit une intercorrélation avec la ligne correspondante de la matrice du signal émis. Il en ressort une matrice du signal reçu filtré, qui est ensuite traitée par le CFAR ligne par ligne, enfin une fois les cibles détectées, le calcul de distance est effectué et le résultat est disponible pour le tracking.

Cette refonte a donc été très bénéfique puisqu'elle a permis de connecter tous les éléments de la chaîne de traitement, elle nous a permis de traiter l'ambiguïté concernant l'association des signaux retours à l'émission qui correspond. Enfin ce format matriciel nous ouvre de nouvelles possibilités comme le calcul Doppler ou encore l'intégration cohérente.

I.5 – Principe du tracking :

Le pistage, ou tracking, constitue la dernière étape du traitement de l'information réceptionnée par l'équipement en ce qui concerne notre projet. Elle intervient après la détection. L'étape de détection met à disposition de l'algorithme de tracking des *plots* : ces derniers doivent être liés à une cible et donc à la piste (*track*) liée à cette dernière. Les pistes sont donc une représentation du trajet de la cible. Le but de l'algorithme de tracking est de mettre à jour ces pistes pour permettre de suivre une cible à l'aide des différentes acquisitions du radar.

On va ici décrire un procédé simplifié et très général de la méthode de pistage^[5]. On décrira ensuite les différentes méthodes et différents algorithmes à disposition pour implémenter le pistage en insistant plus particulièrement sur les deux algorithmes que nous avons décidé d'implémenter pour notre projet : l'algorithme des k plus proches voisins (K-NN) et le filtrage de Kalman.

Tout d'abord, chaque piste (*track*) a son historique de données c'est à dire tous les plots qui lui ont été attribué. Ces derniers sont stockés dans une liste.

On commence le tracking quand les capteurs détectent correctement un point d'intérêt, c'est à dire qu'on prend soin d'éliminer des données parasites pour éliminer les faux départs. On peut se reporter à la section *Détection I.3* pour plus de précisions. Une fois ce point communiqué à l'algorithme de pistage, nous sommes face à 3 situations différentes :

- Le point ne correspond à rien : on le *drop* i.e on décide qu'il n'est pas digne d'intérêt après comparaison avec les données disponibles.
- *Track initiation* : On associe le point à une nouvelle piste et donc au pistage d'un autre objet que l'on ne suivait pas avant.
- *Plot to track association* : l'algorithme reconnaît la donnée comme faisant partie d'une piste existante. On met donc à jour la piste en ajoutant ce nouveau plot à la liste de ses plots.

Pour pouvoir réaliser ces étapes, on effectue une prédiction de la nouvelle position basée sur les dernières données recueillies (pour chaque pistes). Pour pouvoir prédire la nouvelle position, on se base sur le dernier état connu de la cible (position, direction, vitesse, accélération). Puis on compare cette prédiction à notre plot détecté. Pour cela on utilise principalement 4 algorithmes.

I.5.1 – Algorithme des plus proches voisins (K-NN pour K Nearest Neighbours) :

Cette méthode se base sur un principe plutôt simple : on se sert des k plots (dont on sait à quelle piste ils sont associés) les plus proches du dernier plot relevé par l'algorithme de détection pour déterminer à quelle piste il appartient.^[2]

L'algorithme calcule tout d'abord la distance du dernier plot à chaque plot détecté au préalable pour établir la liste des k plus proches voisins de notre point d'intérêt. Dans notre cas, on utilise la distance euclidienne. D'autres distances sont parfois utilisées comme la distance de Mahalanobis.

Puis chacun de ces k plots est classé : dans notre cas, une classe représente une piste. Il nous reste à déterminer à quelle classe le plot d'intérêt appartient. On choisit de lui attribuer la classe majoritaire parmi ses k plus proches voisins. Parce qu'on fait le choix d'un "vote majoritaire" on fixe la valeur de k à un nombre impair (ici 3) pour éviter un cas d'égalité et donc éviter de bloquer l'algorithme qui ne dispose pas d'élément pour privilégier une classe plutôt que l'autre.^[7] Pour une première version de l'algorithme K-NN, se reporter à l'annexe.

I.5.2 – Probabilistic data association filter PDAF

Ce type de filtre sert uniquement à associer un plot à un piste. On ne peut pas s'en servir pour initier une piste ou pour la terminer. Comme son nom l'indique, on se base ici sur une approche statistique. On recherche les plots qui ont le plus de chances (statistiquement parlant) de correspondre à la position prévue. Pour cela, on met en place une zone de validation autour du point estimé, c'est à dire une zone dans laquelle chaque plot qui s'y trouve peut correspondre à la cible (cela permet de réduire l'espace dans lequel on fait la recherche). Parmi tous les plots se trouvant dans cette zone, il faut alors trouver celui qui vient bien de la cible et le rattacher la piste correspondante.^[8]

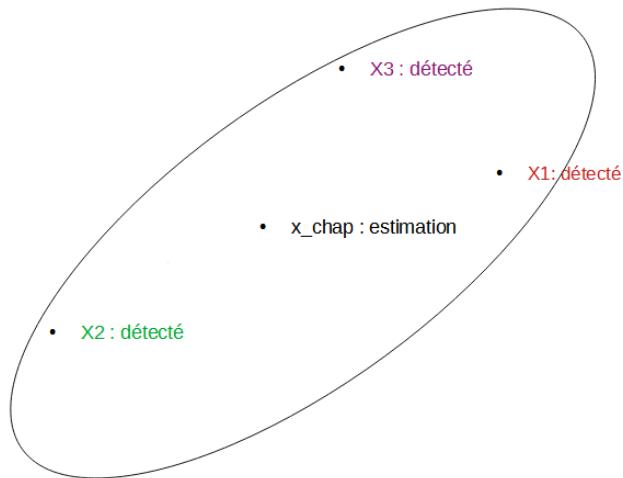


Figure 31 : Zone de validation autour de la position estimée

Le filtre, dans sa version la plus simple, va alors calculer l'espérance de chaque plot détecté et chercher à minimiser l'erreur quadratique moyenne. Chaque plot possède un poids différent. Le plot avec l'erreur quadratique moyenne la plus faible sera retenu pour être intégré à la piste de la cible. Ce type d'algorithme se montre efficace quand on cherche à suivre une cible seulement au milieu d'un nuage de données^[9]

I.5.3 – Multiple Hypothesis Tracking (MHT)

L'algorithme de pistage avec hypothèse multiple est une version plus avancée du PDAF. Avec ce type d'algorithme, une piste peut être mise à jour par plusieurs données. Il en résulte un schéma (appelé « arbre ») qui bifurque dans plusieurs directions possibles créant plusieurs pistes possibles, plusieurs alternatives. Chaque arbre correspond à une cible. L'algorithme calcule ensuite la probabilité de chaque branche et sélectionne la combinaison la plus probable, offrant ainsi la trajectoire de la cible.^[10]

Cet algorithme prend en compte toutes les mesures transmises par l'étape de détection. L'ajout d'une mesure ne fait qu'étendre les différentes branches de l'arbre comme on peut le voir sur cette représentation graphique tirée des travaux de A. Amditis et al.^[10]

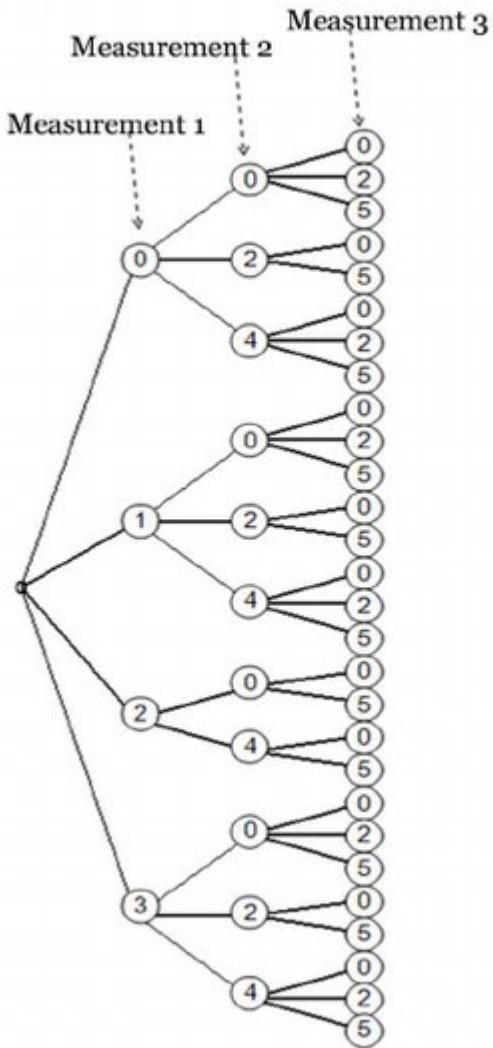


Figure 32 : Arbre formé par les différentes hypothèses^[11]

Le fait que l'algorithme conserve toutes les pistes possibles fait qu'il est particulièrement intéressant de l'utiliser quand le mouvement de la cible traquée est trop erratique.

I.5.4 – Filtrage de Kalman

Le filtrage de Kalman est le second algorithme que nous tenterons d'implémenter pour analyser nos acquisitions et tenter de pister une cible. C'est une des méthodes de filtrage les plus utilisées aujourd'hui. Un filtre de Kalman possède deux aspects : un concernant la *prédiction* et l'autre concernant la *correction* des mesures.

Contrairement au filtre précédent, celui-ci est récursif, on ne prend en compte que la valeur mesurée précédente.^[7] Cette valeur qui peut être bruitée est corrigée à l'aide des mesures corrigées précédentes, pour obtenir une valeur plus proche de la valeur réelle, puis à l'aide de cette dernière, on prédit la nouvelle position au temps présent. Le filtre a l'avantage notable de corriger son erreur au cours de son utilisation.

Pour une première version de l'algorithme soutenant le filtrage de Kalman, se reporter à l'annexe. Comme pour l'algorithme K-NN, cette méthode a été implémentée “seule” c'est à dire sans avoir un jeu de données d'entrée.

I.5.5 - Implémentation

Au cours de la deuxième étape du projet, nous avons développé principalement d'algorithme de tracking se basant sur les K plus proches voisins. Pour se faire, nous avons dû créer un programme mettant en œuvre les trois axes de l'algorithme K-NN évoqués plus haut. C'est à dire l'initialisation d'une piste, le suivi d'une cible par la mise à jour de sa piste et la suppression de piste non mise à jour depuis un certain nombre d'acquisition (ici 4 est notre palier). De plus, le travail de tracking s'effectue en post traitement et non pas en temps réel, les algorithmes s'effectuent donc en mode "pas à pas". Pour chaque donnée sortie du CFAR, nous faisons tourner l'algorithme de tracking.

En un premier lieu, nous avons implémenté un algorithme travaillant avec en entrée des coordonnées cartésiennes (x, y). Nous avons créé un code capable de suivre une cible et d'initialiser une seconde piste dont voici les résultats d'exécution après un test unitaire. Nous l'avons testé sur un jeu de données "propres" c'est à dire inventées dans ce but. Il ne s'agit pas d'une sortie du CFAR. Les coordonnées de test sont les suivantes pour les 2 cas traités :

```
if __name__ == "__main__":
    donnee = [5, 5]
    donnee_eclatée = [-20, -20]
    k = 3 # nombre de voisins souhaité
    piste_1 = [[7, 7, 1],
               [8, 7, 1],
               [1, 2, 1],
               [-2, 2, 1],
               [2, 2, 1]
              ]
    piste_2 = [[2, 1.8, 2],
               [6, 5, 2],
               [8, 6, 2],
               [6, 6.4, 2],
               [0, 3, 2]]
    piste_3 = [[0, 2, 3],
               [1, 3, 3],
               [1.5, 4, 3],
               [3, 2, 3],
               [4, 3, 3]]

    suivi_2_cible(donnee, k, piste_1, piste_2)
    |suivi_1_cible(donnee_eclatée, piste_3)
```

Figure 33 : Jeu de données « propres »

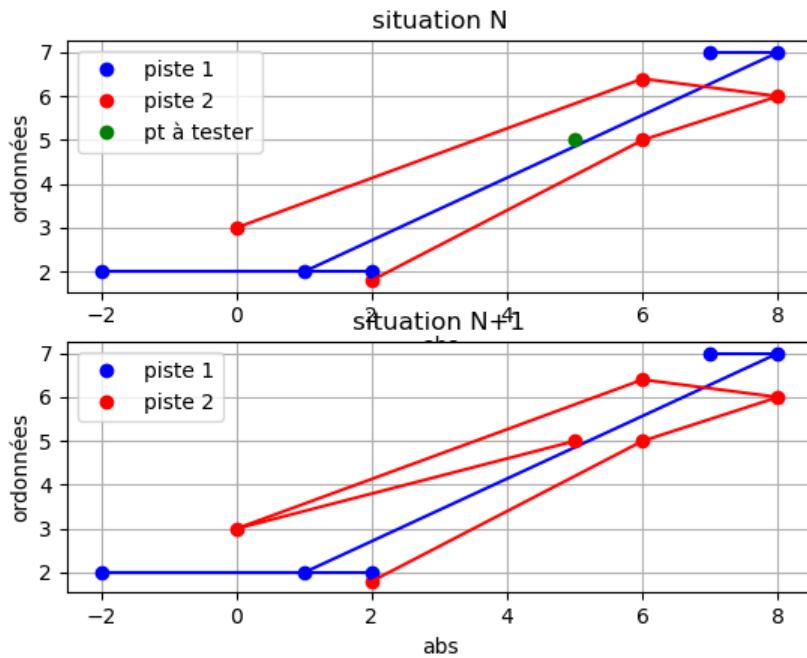


Figure 34 : Un pas de l'algorithme de tracking en 2D. Affectation d'un plot à la piste adéquate avec 3 voisins.

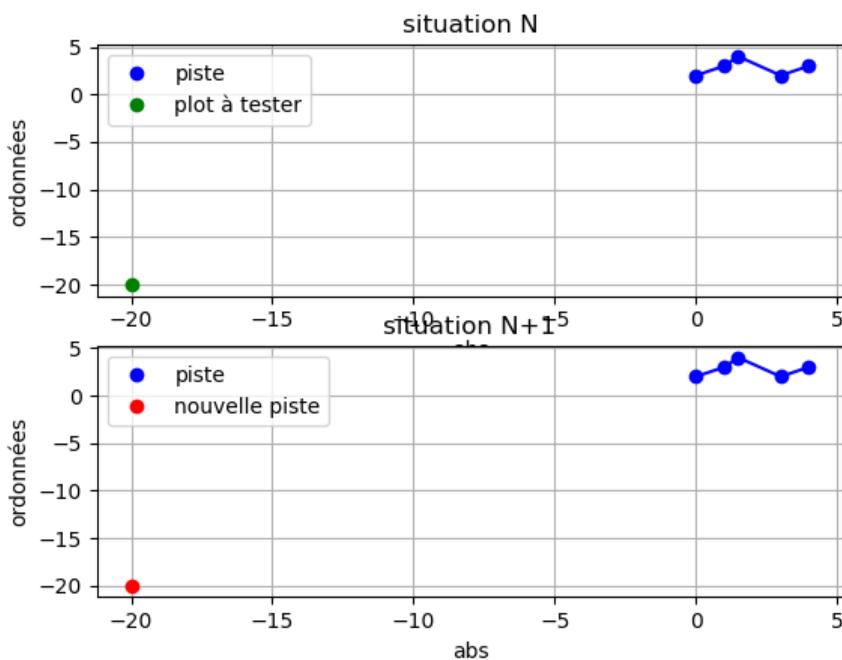


Figure 35 : Initialisation d'une nouvelle piste avec un plot délibérément placé au-dessus du seuil de l'algorithme de tracking 2D

Pour déterminer si l'algorithme initialise une nouvelle piste ou pas selon les données passées en entrée, nous nous basons sur un seuil. Ce seuil est déterminé grâce à une estimation de la vitesse de l'objet suivi et de la durée entre chaque acquisition. Si la distance au plot précédemment traité par l'algorithme est supérieure à ce seuil, il est décidé que le plot ne peut physiquement pas être une détection de la cible suivie initialement. On initialise donc une nouvelle piste.

Ce seuil doit être déterminé selon le type de cible suivi. En effet il sera différent si l'on suit un piéton ou une voiture. Lors des tests unitaires avec les données d'entraînement, nous avons imposé le seuil à 5 de manière totalement arbitraire. Pour le test d'intégration de l'algorithme, il sera créé une fonction prenant en argument la vitesse de déplacement de notre objet ainsi que le temps entre deux acquisitions pour nous renvoyer un seuil adapté à notre manipulation.

Néanmoins, lors de la mise en place de notre manipulation, le matériel disponible ne nous permettait pas d'avoir, à terme, un jeu de coordonnées 2D pour le traitement. En effet nous ne disposons pas d'antennes tournantes. Nous avons alors codé une deuxième version de l'algorithme K-NN qui prend en entrée seulement une coordonnée cartésienne correspondant à un déplacement sur l'axe des ordonnées et reflétant le déplacement de notre réflecteur. Là encore l'algorithme est capable de suivre une cible et d'initialiser une nouvelle piste.

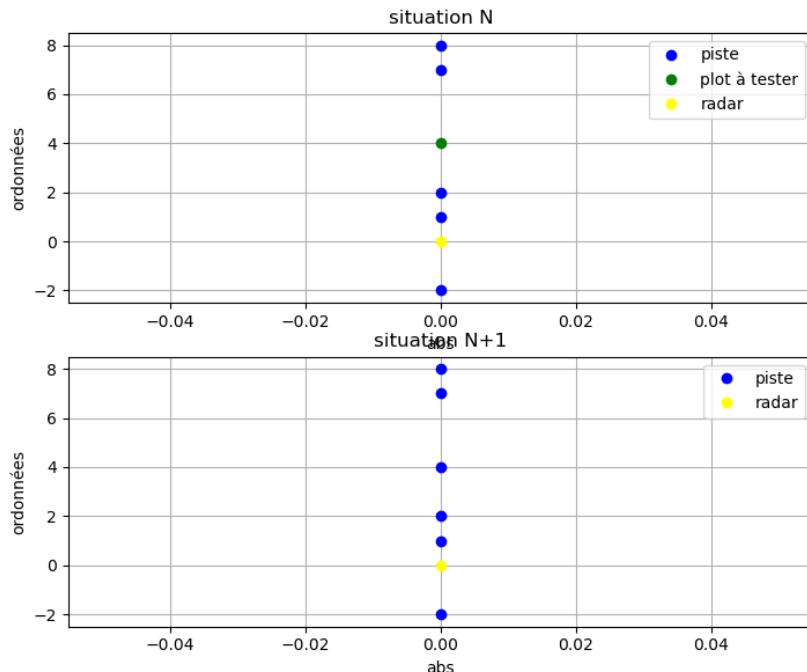


Figure 36 : Mise à jour de la piste avec l'algorithme de tracking en 1D avec 3 voisins. Points de la piste initialisés au préalable et plot avec une coordonnée cohérente. (Test unitaire)

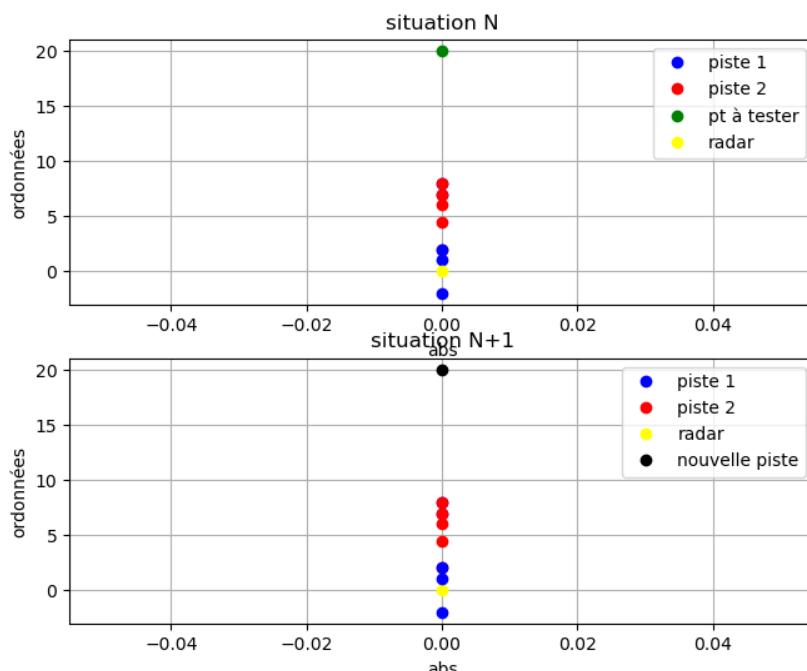


Figure 37 : initialisation d'une nouvelle piste avec 3 voisins avec un plot délibérément placé loin du radar. Les 2 pistes présentes ont été initialisées avec un jeu de données "propres" (test unitaire)

Les tests unitaires effectués sont semblables à ceux mis en place pour le tracking 2D. Pour le suivi de la cible, nous avons en entrée une piste de 5 plots et une donnée. Nous l'avons testé avec deux types de données différents : une donnée "correcte" permettant de vérifier que le suivi s'effectue correctement et une donnée "éloignée" permettant de vérifier la capacité de l'algorithme à initialiser une piste.

Puis nous avons testé l'algorithme en sortie du CFAR avec la matrice des distances donnée par ce dernier. Les données sont celles des acquisitions du 16 mars 2021. Nous avons dû pour cela modifier les premières étapes du suivi. En effet, les algorithmes étaient précédemment testés avec en paramètre d'entrée une piste et une donnée. Pour le cas réel, nous ne disposions pas de la piste. Nous avons donc appliqué une fois l'algorithme avec la première donnée jouant le rôle de piste et les deux suivantes comme données « classiques ». Nous avons donc passé le nombre de voisins à 1. Puis avec cette piste de trois éléments, nous avons appliqué l'algorithme de même manière que lors des tests unitaires avec 3 voisins.

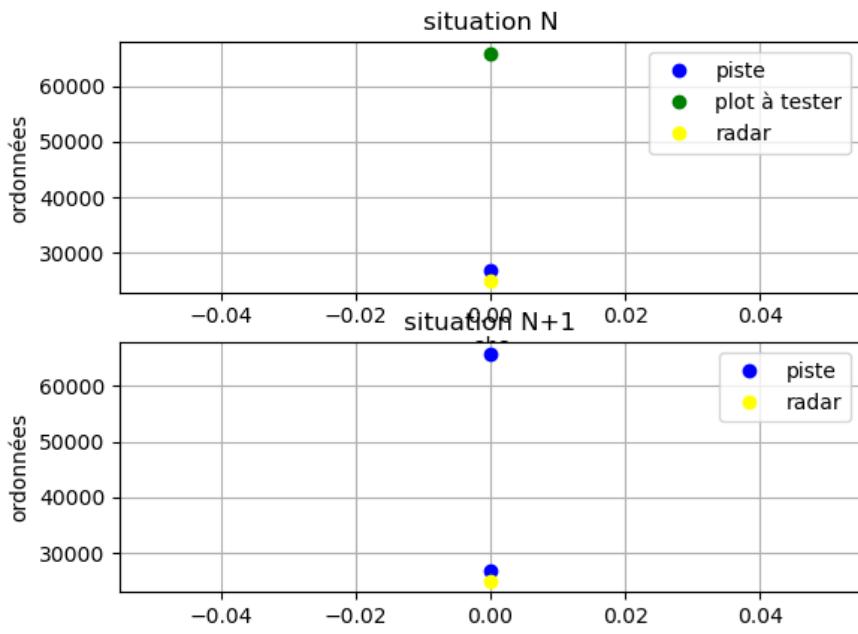


Figure 38 : Algorithme en sortie du CFAR, sur acquisition

```
[list([26832.1, 62832.1, 65832.0999999999, 104832.1, 145332.0, 179831.5999999998, 182831.6, 185831.6])
 list([25332.60000000002, 28332.60000000002, 64332.3, 67332.3, 100332.0, 104832.1, 143831.8, 146831.5, 182831.1999999998, 185831.1999999998])
 [[104832.1], [145332.0], [179831.5999999998], [182831.6], [185831.6]])
on effectue 2 tours d'algo
let's start
1 ème tour avec 1 piste
le plot appartient à la piste
piste mise à jour: [[26832.1, 6], [65832.0999999999, 6]]
le nouveau point est à : 40832.0999999999 m du radar
le point c'est déplacé de: 38999.9999999999 m entre son emplacement précédent et son emplacement actuel
```

Figure 39 : Initialisation avec les données du CFAR

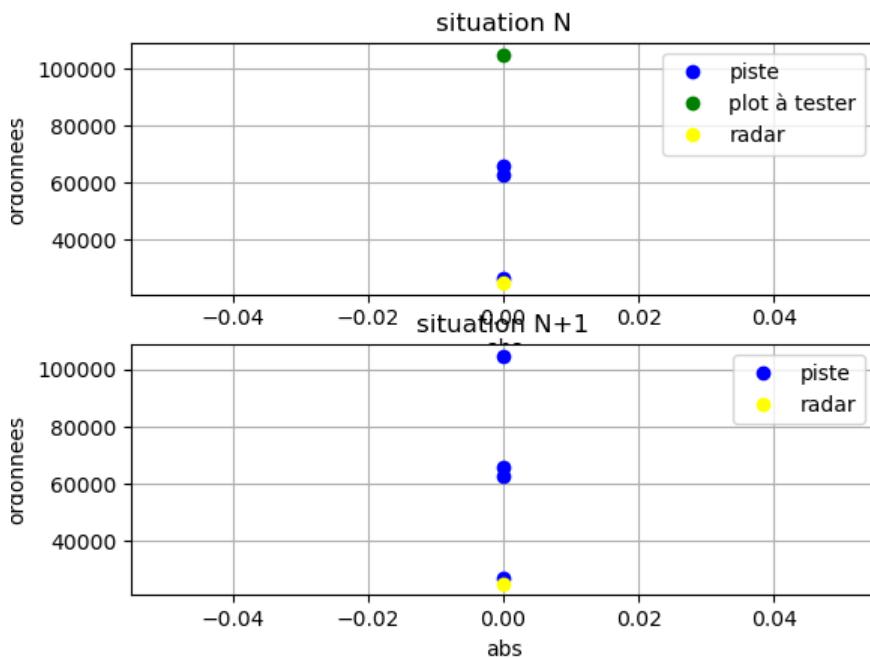


Figure 40 : Algorithme en sortie du CFAR, sur acquisition

```

on effectue 1 tours d'algo
let's start
1 ème tour avec 1 piste
le plot appartient à la piste
piste mise à jour: [[26832.1, 6], [65832.0999999999, 6], [62832.1, 6], [104832.1, 6]]
le nouveau point est à : 79832.1 m du radar
le point c'est déplacé de: 42000.0000000001 m entre son emplacement précédent et son emplacement actuel
piste 1 updated [[26832.1, 6], [65832.0999999999, 6], [62832.1, 6], [104832.1, 6]]
une deuxième piste ? []

```

Figure 41 : Continuation de l'algorithme de suivi

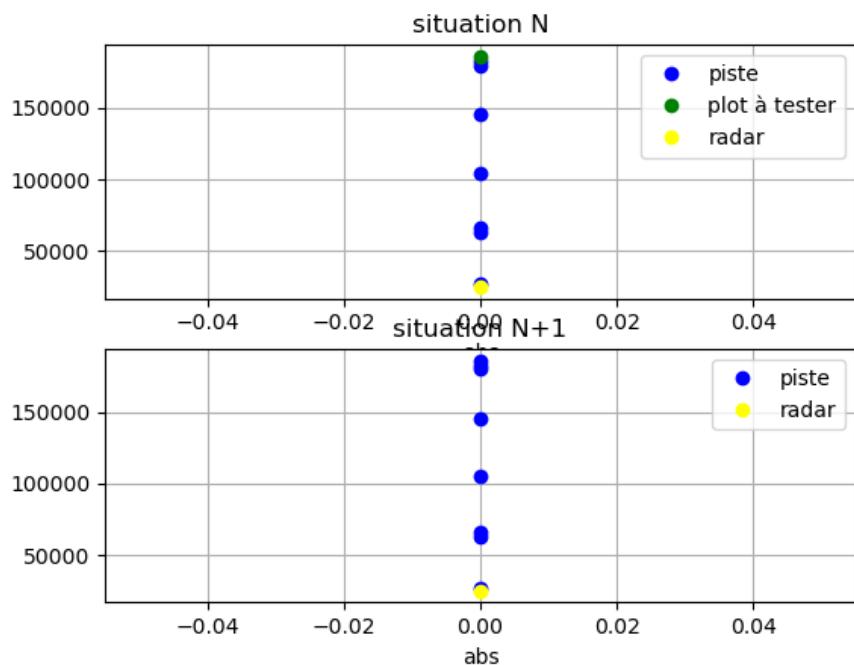


Figure 42 : Algorithme en sortie du CFAR, sur acquisition

```
1 ème tour avec 1 piste
le plot appartient à la piste
piste mise à jour: [[26832.1, 6], [65832.0999999999, 6], [62832.1, 6], [104832.1, 6], [145332.0, 6], [179831.5999999998, 6], [182831.6, 6], [185831.6, 6]]
le nouveau point est à : 160831.6 m du radar
le point c'est déplacé de: 3000.0 m entre son emplacement précédent et son emplacement actuel
```

Figure 43 : Fin de l'algorithme

```
[[26832.1, 6], [65832.0999999999, 6], [62832.1, 6], [104832.1, 6], [145332.0, 6], [179831.5999999998, 6], [182831.6, 6], [185831.6, 6]]
```

Figure 44 : Piste finale

On constate que tous les points ont été affecté à la piste 6 ce qui vérifie nos observations. Cependant, l'algorithme possède encore quelques faiblesses et donc des points d'amélioration. En effet, ce dernier est capable d'initialiser une seconde piste (ce qui était dans le cahier des charges) mais il n'est pas capable à partir de l'initialisation de la seconde piste de suivre les deux cibles.

II – Les principes de l'ingénierie système :

Comme nous sommes dans la première partie du projet système, nous sommes ici dans la phase de *conception* du cycle de vie du système. Autrement dit, dans la première partie du cycle en V. Nous allons chercher à exposer dans cette partie les exigences, l'architecture fonctionnelle et l'architecture physique qui ont soutenus la conception de notre projet.

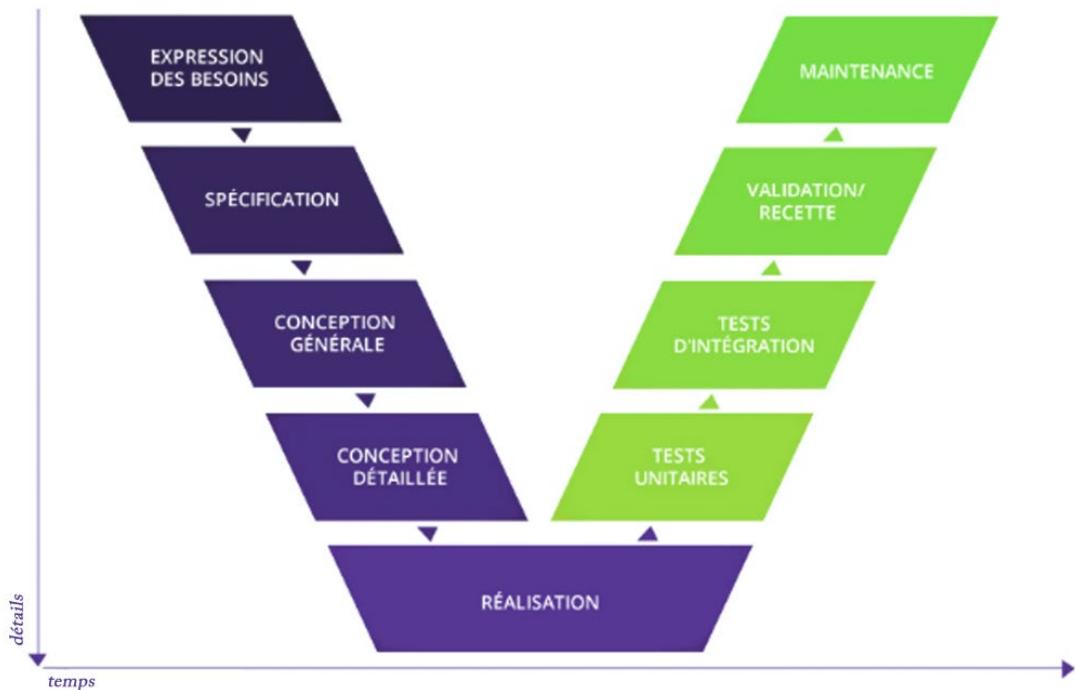


Figure 45 : cycle en V

II.1 – Exigences et architecture fonctionnelle

Fonction	Description
FP : Détection	Déetecter un objet immobile ou en mouvement et rendre compte de sa vitesse radiale. Utiliser l'algorithme CFAR
FS : Matière	Rapporter la matière de l'objet détecté par le radar
FC : Filtrage	Réduire le bruit environnant pour analyser correctement l'écho d'ondes électromagnétiques/ Augmenter le rapport signal sur bruit
FS : Simulation des signaux	Produire des signaux proches de ceux obtenus avec la carte d'acquisition
FP : Suivi de cible	Utiliser la technique du beamforming pour orienter électroniquement le lobe d'antenne en fonction du mouvement de la cible
FS : Distinguer	Trouver la distance minimale nécessaire pour détecter deux objets proches
FS : Performance	Trouver la taille minimale requise pour que l'objet soit détecté par l'antenne. Trouver les distances minimales et maximales de l'objet à l'antenne

Tableau 1 : Tableau des fonctions du système

II.2 – Architecture physique

La conception de l'architecture physique du système se base sur les exigences exprimées précédemment et sur des principes physiques. Chaque composant est sélectionné pour répondre à un besoin précis. L'équipe encadrante a imposé une liste de matériel à utiliser. Il nous faut donc comprendre le rôle de chaque élément puis effectuer la connectique entre ces derniers.

Comme exposé dans la partie I.1, la base du fonctionnement d'un radar est l'émission et la réception d'une onde. Lors du premier sprint nous avons donc cherché à mettre en place une chaîne de composants pouvant générer une onde hyperfréquence, nous avons aussi réfléchi au moyen d'acquérir cette dernière.

Le radar est un radar hyperfréquence. On doit donc générer une onde ayant une fréquence de l'ordre de la dizaine de GHz (giga hertz). On choisit l'onde et sa fréquence en fonction du milieu de propagation de l'onde et de l'utilisation du radar. On ne veut pas un radar très longue portée qui soit capable de traverser plusieurs milieux avant de trouver sa cible (qui nécessiterait une longueur d'onde de l'ordre du mètre) mais un radar courte portée pouvant détecter une cible en extérieur et visible : le choix d'une fréquence de 10 GHz est donc cohérent avec notre utilisation. Le signal que nous cherchons à émettre à terme est composé de *pulses*. Avant d'en arriver là, nous cherchons d'abord à générer et modifier correctement un signal simple de type sinusoïde. Cette dernière est générée à une fréquence relativement basse (un peu moins d'un GHz). Il nous faut donc augmenter considérablement la fréquence, ce qui est fait à l'aide d'un Up Converter. Il n'y a ici aucune perte d'informations car l'augmentation de la fréquence ne modifie pas la largeur de bande (500 MHz) qui contient toutes les informations protégées par le signal. Bien sûr pour analyser le signal renvoyé par l'objet traqué par le radar, on diminuera la fréquence pour revenir à l'ordre de grandeur des valeurs de départ.

L'ensemble des composants, leur rôle ainsi que les connections entre eux sont exposés dans le tableau et la figure ci-dessous :

Composant	Fonction dans la chaîne	Explications
Générateur 100B	Génération de l'onde	Génération de la sinusoïde
Générateur 100A	Modulation en fréquence du signal émis	Va synthétiser un signal complexe, à bande plus large (500 MHz) et augmenter la fréquence du signal (1GHz). On travaille avec des fréquences hautes mais le générateur 100A ne suffit pas à atteindre nos fréquences de travail
Splitter	Intermédiaire	Le signal d'arrivée est dupliqué. Cela permet de lui appliquer 2 traitements différents
Up converter	Augmentation de la fréquence du signal avant de le transmettre à l'antenne radar	On atteint les très hautes fréquences : 10/15 GHz
Antenne	Emission et réception de l'onde	L'antenne a pour but d'envoyer l'onde sur l'objet à détecter et de

		réceptionner l'onde émise par l'objet
Down converter	Diminution de la fréquence du signal revenant du radar	Le down converter fait partie de la chaîne chargée de réceptionner l'onde en provenance du radar. On revient donc à nos fréquences de base (1GHz) pour analyser le retour du radar
Oscilloscope	Analyse du signal	Analyse temporelle du signal envoyé vers l'antenne
Analyseur de spectre	Analyse du signal	Analyse spectrale du signal envoyé vers l'antenne. Cependant l'analyseur ne peut pas être utilisé avec de très hautes fréquences

Tableau 2 : Tableau des composants

La chaîne que nous avons mise en place pour l'émission de l'onde est la suivante :

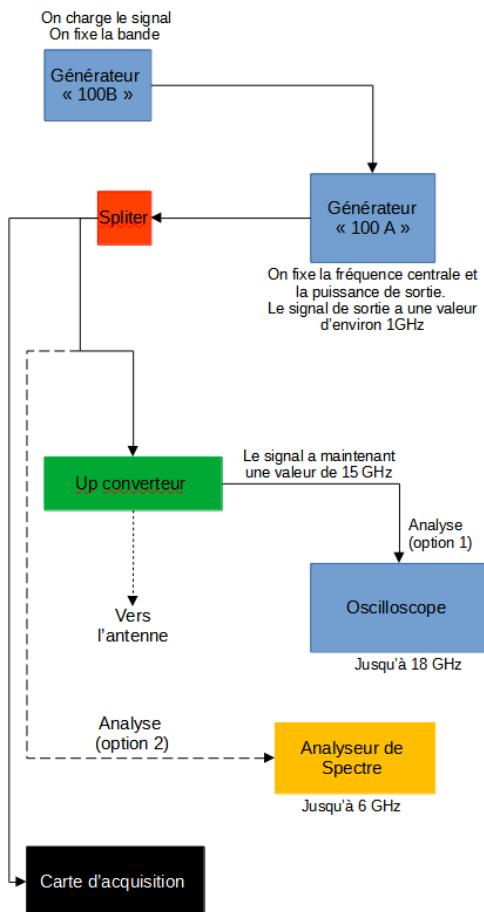


Figure 46 : Chaîne mise en place pour générer le signal

La mise en place de la chaîne doit s'accompagner de différents tests sur les performances de chaque élément. Ces tests unitaires sont nécessaires pour établir les caractéristiques et performances de chaque composant de la ligne et sont donc cruciaux pour pouvoir déterminer les caractéristiques du radar mais aussi sa performance globale.

Les premiers tests unitaires ont été entamés lors du sprint numéro 1 (06/10/2020 au 20/10/2020). Ces derniers devaient être plus poussés mais en raison du confinement national, nous n'avons pas pu avoir accès au matériel pour les poursuivre.

Algorithme(s) concerné(s)	Type de test	Principe
Tracking – 2D	Unitaire	Avec un jeu de données "propre" montrer que l'algorithme est capable d'associer un plot à la piste correspondante en utilisant 3 voisins
Tracking – 2D	Unitaire	Avec un jeu de données "propre" montrer que l'algorithme est capable de discriminer un plot et d'initialiser une nouvelle piste
Mise en forme du signal reçu	Unitaire	Grâce aux grandeurs caractéristiques du signal, vérifier que l'algorithme est capable de mettre le signal sous forme matricielle sans erreur.
Filtrage	Unitaire	A l'aide de signaux simple, vérifier la cohérence du filtrage effectué.
Détection - CFAR	Unitaire	A l'aide d'un signal généré arbitrairement, vérifier que l'algorithme est bien en mesure de détecter les pics.
Détection - CFAR	Unitaire	A l'aide d'un signal brut, vérifier que l'algorithme est bien en mesure de détecter les pics d'importance.
Détection - CFAR	Unitaire	A l'aide d'un signal réel et filtré, vérifier que l'algorithme est robuste à forme particulière de ce signal.
Calcul de distance	Unitaire	Sur des valeurs générées, vérifier que l'algorithme calcule la bonne valeur de la distance cible – radar.
Mise en forme et filtrage	Intégration	Bonne interprétation par le filtrage de la mise en forme effectuée.
Filtrage, détection et calcul de distance	Intégration	Bonne interprétation par le CFAR du signal filtré, puis du calcul des distances cibles – radar.

Tracking – 1D	Unitaire	Avec un jeu de données “propre” montrer que l’algorithme est capable d’associer un plot à la piste correspondante en utilisant 3 voisins
Tracking - 1D	Unitaire	Avec un jeu de données “propre” montrer que l’algorithme est capable de discriminer un plot et d’initialiser une nouvelle piste
Tracking - 1D	Intégration	Suivi de cible concluant avec jeu de données issu du CFAR. Le seuil a été imposé pour coller aux observations.

Tableau 3 : Liste des tests unitaires effectués sur les algorithmes

III – L’application de la méthode Agile au sein de l’équipe du projet ObsHyper :

Le projet s'est articulé en trois temps lors de ce semestre. La première phase a été pour nous un temps d'appropriation du sujet et des méthodes agiles qui allaient nous permettre de gérer l'organisation et le fonctionnement de l'équipe. Cette phase du projet a été ponctuée de nombreux rendez-vous avec les encadrants afin d'orienter l'équipe et de la familiariser avec le matériel mis à disposition par l'ENSTA Bretagne. En effet face à l'angle d'approche très large du sujet et la quantité d'informations, les encadrants ont pris soin de nous accompagner sur le plan théorique afin d'être sûrs que nous ayons les bases nécessaires à l'implémentation de ce projet. Cette phase a duré jusqu'au 6 octobre 2020. De plus, nous avons pendant cette période, nous avons commencé à mettre en place la méthode Agile dont nous nous servons pour gérer notre équipe. Le rôle de Product Owner a été attribué à Pierre-Olivier et celui de Scrum Master à Emilie. Nous avons aussi commencé à utiliser régulièrement l'outil Trello pour suivre l'avancement des tâches que nous devions remplir, ceci afin d'être prêts pour commencer les sprints.

Au 6 octobre 2020, nous avons lancé le premier sprint. Dès le commencement du projet, les tâches respectives de chacun des membres de l'équipe ont été choisies par affinités et cela s'est poursuivi lors de la répartition des tâches du sprint 1. Chacun d'entre nous représentant un maillon de la chaîne censée nous mener au produit fini. Nous sommes dépendants du travail de celui qui nous précède dans ladite chaîne. A la fin de sprint, nous avions mis en œuvre la chaîne permettant de générer le signal à destination de l'antenne et avions commencé à exploiter les résultats d'acquisitions simples. Il a été mis en avant, au cours de ces premières semaines, que notre force ici, est notre bonne entente et une aisance de communication non dissimulable, d'autant que l'intérêt que chacun porte personnellement à ce projet est manifeste. Ce qui a été rapporté lors de la première réunion avec notre encadrante *Méthode Agile* Pascale Gautron.

Younouss, jusqu'en octobre notre partenaire, a dû nous quitter, faute de pouvoir rejoindre l'école, nous avons sans peine redistribué sa charge de travail équitablement entre nous et avons poursuivi nos activités à quatre. Or, le 20 octobre 2020, nous avons été pour la deuxième fois confronté à un confinement et à la mise en place des cours à distance. Après une semaine de réadaptation et de consultation avec nos encadrants, nous avons lancé le sprint 2 avec un mode de fonctionnement plus rigide. L'équipe s'est alors articulée comme suit : Julien et Pierre-Olivier ont travaillé, à partir du confinement, sur l'acquisition des données depuis un fichier type récupéré sur les outils de l'école avant le confinement, et le traitement du signal afin de les rendre exploitables par Jany qui travaille sur un algorithme de détection de cible, suivi de près par Emilie qui travaille sur le suivi de ces mêmes cibles. Les recherches pour établir un état de l'art dans ces différents domaines et les premiers développements des algorithmes nécessaires se sont poursuivis jusqu'à la fin du sprint 3 (01/12/2020). Puis pour le dernier sprint du semestre (le sprint 4), nous avons décidé de nous concentrer sur l'écriture du rapport de projet et la mise en forme des différents attendus.

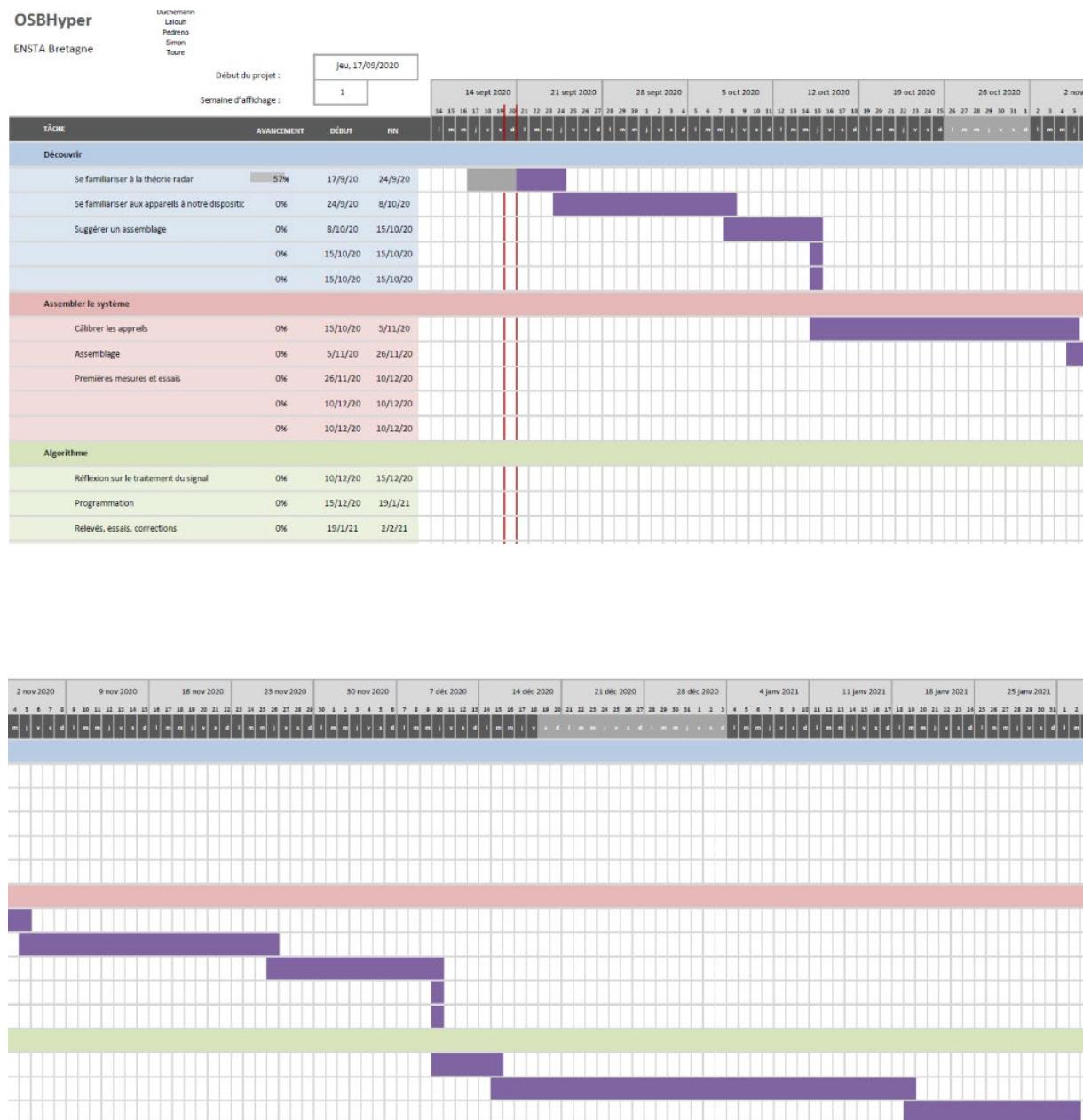
Pour autant, et tant que c'est possible, nous avons tenu à ce que chacun d'entre nous soit capable de comprendre les travaux des autres, c'est pourquoi, au-delà des tâches assignées à proprement parler, nous avons pris soin de nous intéresser en profondeur aux travaux de nos pairs, afin de profiter, à terme, d'un maximum d'agilité.

Le suivi des sprints s'est fait à l'aide des outils Mahara et Trello dont voici les accès :

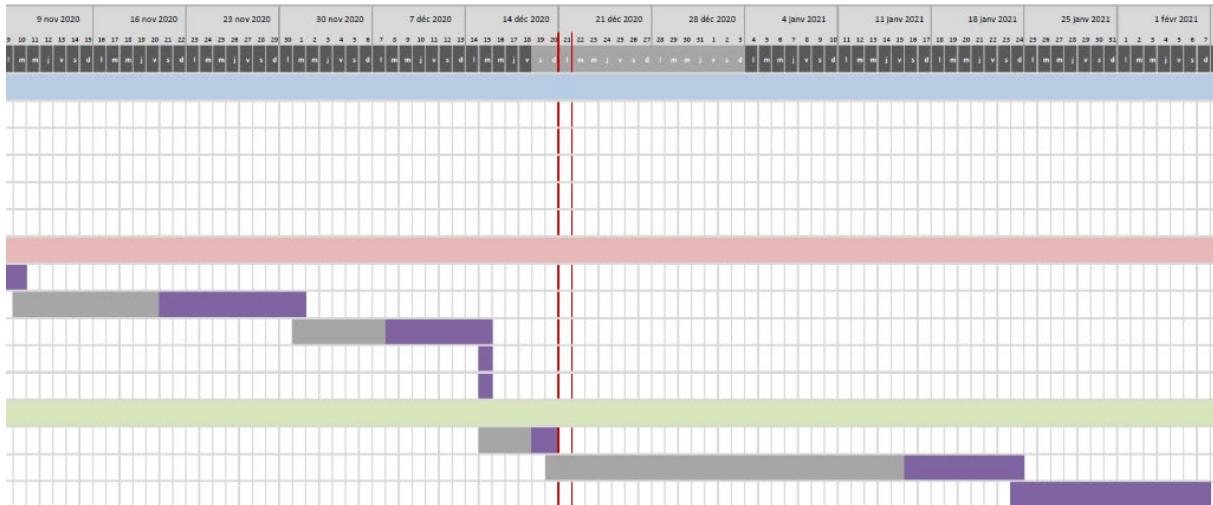
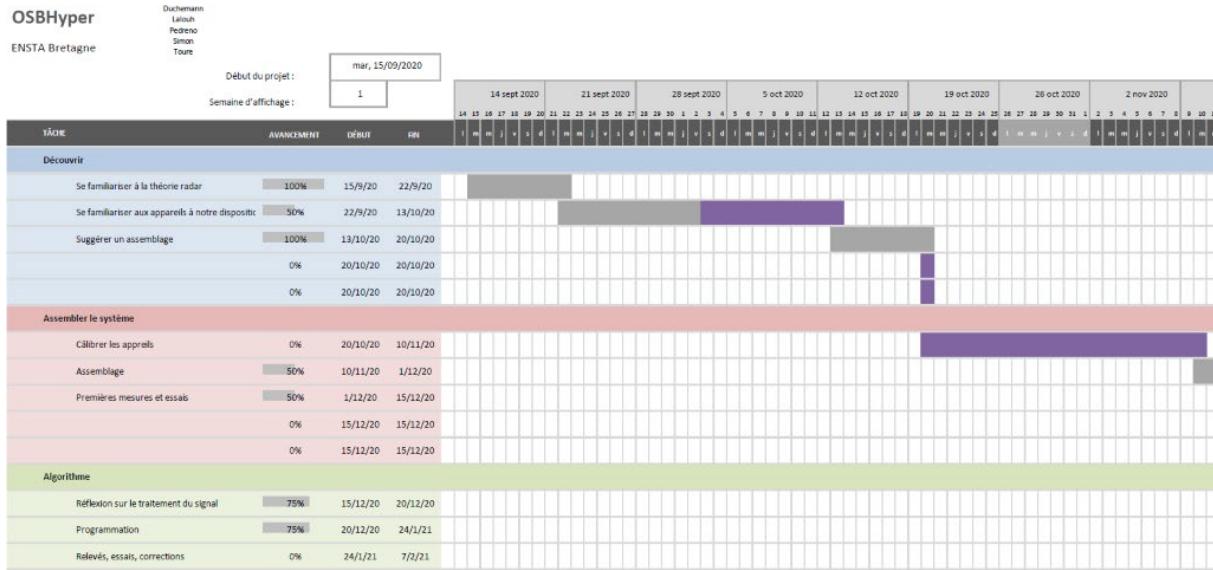
- Mahara : <https://mahara.ensta-bretagne.fr/view/groupviews.php?group=346>
- Trello : <https://trello.com/b/YCVPYZwc/obshyper>

L'éloignement géographique a mis l'équipe au défi, notamment en ce qui concerne la communication et la compréhension des attendus. L'équipe s'est donc tournée vers les réseaux sociaux et les plateformes de communication de type Messenger pour garder le contact. Nous avons

cependant conservé Teams pour communiquer avec nos encadrants et pour faire un point sur l'avancement de chacun toutes les deux semaines.

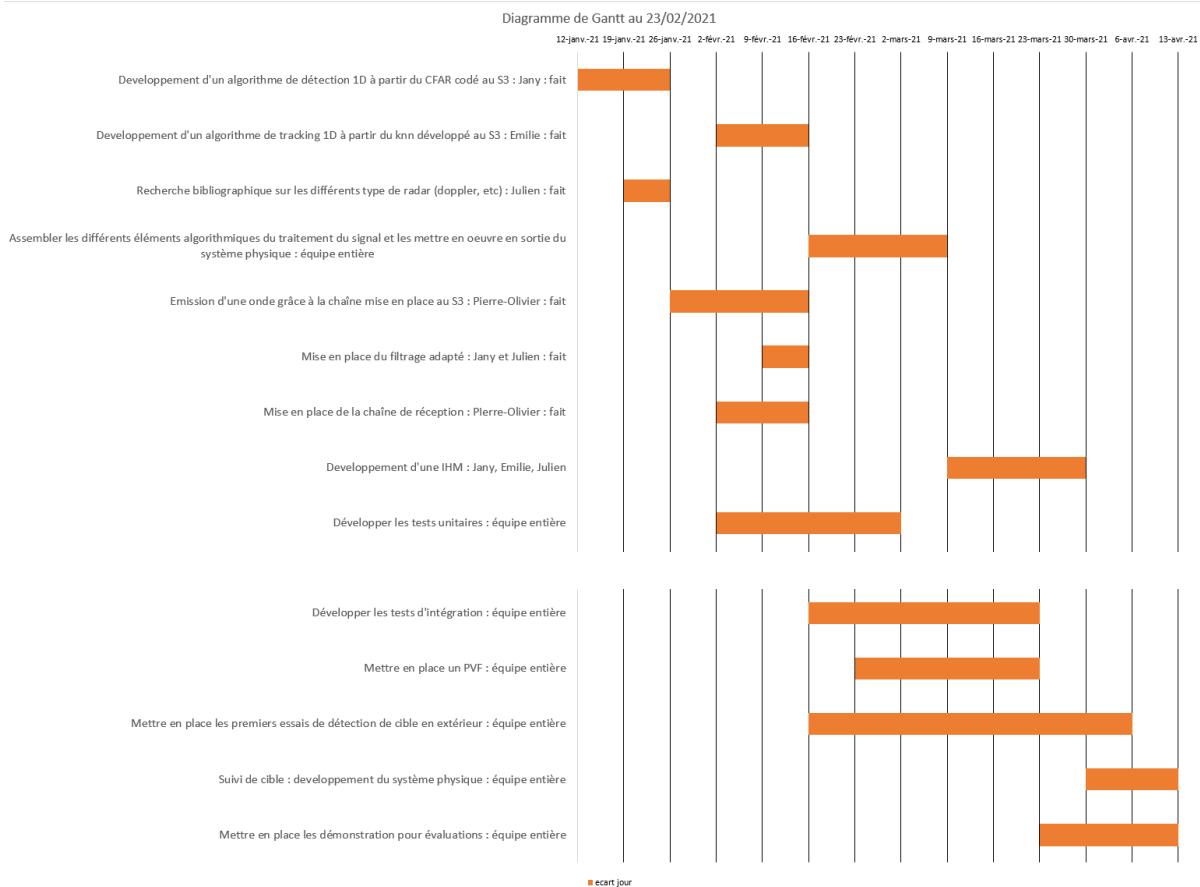


Figures 47 et 48 : Diagramme de Gantt en début de projet



Figures 49 et 50 : Diagramme de Gantt aux vacances de Noël

Le commencement de l'acte 2 du projet a impliqué la mise à jour de notre diagramme de Gantt. Nous avons revu certaines de nos échéances et rajouté des objectifs.



Figures 51 et 52 : Diagramme de Gantt du 4^e semestre

De plus, face aux faiblesses dans notre communication lors du premier acte du projet, faiblesses mises en avant par le confinement, nous avons repensé notre façon de communiquer. Nous avons mis en place un GitHub pour partager nos codes de manière plus efficace. Nous avons aussi instauré des mélées bihebdomadaires via la plateforme Messenger. Ceci afin de débriefer les avancées de la semaine de travail (et intégrer le travail fait sur temps personnel) et de décider sur quels sujets nous allions travailler la semaine suivante.

En complément, comme peu de personne trouvait le tableau en ligne Trello utile, nous avons décidé de créer un fichier partagé sur Teams qui rassemble les contenus de chaque sprint. Il s'est avéré que les gens le consultaient plus facilement comme il se trouvait au même endroit que tous les autres fichiers (comptes-rendus, codes, ébauches de rapport). Chaque membre de l'équipe est identifié par un code couleur.

Organisation des sprints de l'acte 2 du projet

Code de répartition : Jany Lahlouh Pierre-Olivier Pedreno Emilie Simon Julien Duchemann

Sprint 4.1 : 12 janvier au 26 janvier

A faire	En cours	Fait
	Manipulations du matériel	Planification rendez-vous encadrants
	Finition de l'algorithme des knn en 2D	Mise à jour du diagramme de Gantt
	Recherches sur caractéristiques du radar et critère de performance	(re)prise en main du matériel

	(résolution, ambiguïté) selon les types de radar (doppler/etc.) : résolution convertisseur AN, résolution horizontale radar, distance minimale de sondage, ambiguïté sur la distance	
		Coordonner les entrées et sorties des différents algorithmes (équipe entière)
		Recherche autocorrélation, corrélation, compression d'impulsion, filtrage adapté, signal chirp

Tableau 4 : Sprint 4.1

Sprint 4.2 : 02 Février au 16 Février

A faire	En cours	Fait
	Manipulations du matériel	Finition de l'algorithme des knn en 2D
	Mise en place de la stratégie d'initialisation de piste et de suivi de 2 cibles pour l'algorithme de tracking des knn en 1D	Recherches sur caractéristiques du radar et critère de performance (résolution, ambiguïté) selon les types de radar (doppler/etc.) : résolution convertisseur AN, résolution horizontale radar, distance minimale de sondage, ambiguïté sur la distance
	Création d'une bibliothèque de fonctions de calcul pour la future validation / spécification du système	Mise en place du filtrage adapté (Jany et Julien)
		Modification de la sortie du CFAR, détermination de la distance cible-radar : sortie 1D
		Modification de l'entrée de l'algorithme de tracking knn : entrée 1D
		Production, émission et acquisition d'un signal chirp dans une pièce fermée 30 cm du mur
		Création du GitHub

Tableau 5 : Sprint 4.2

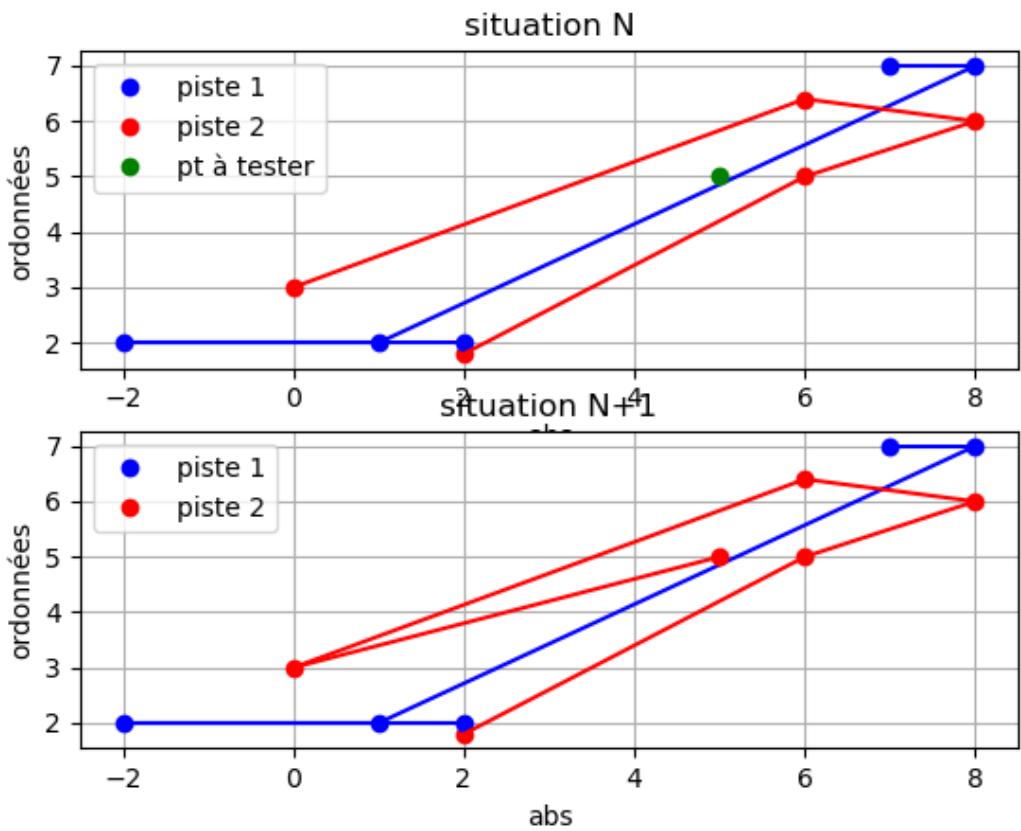


Figure 53 : Sprint 4.2 – Résultat KNN graphique

```
C:\Users\emili\anaconda3\python.exe C:/Users/emili/OneDrive/Bureau/2A/0BShyper/traking/codes/knn.py
voisins: [[6, 5, 2], [6, 6.4, 2], [7, 7, 1]]
le plot appartient à la piste: 2
[[[7, 7, 1], [8, 7, 1], [1, 2, 1], [-2, 2, 1], [2, 2, 1]], [[2, 1.8, 2], [6, 5, 2], [8, 6, 2], [6, 6.4, 2], [0, 3, 2], [5, 5, 2]]]
piste 1 : [[7, 7, 1], [8, 7, 1], [1, 2, 1], [-2, 2, 1], [2, 2, 1]]
piste 2 : [[2, 1.8, 2], [6, 5, 2], [8, 6, 2], [6, 6.4, 2], [0, 3, 2], [5, 5, 2]]
le nouveau point est à : 7.0710678118654755 m du radar
le point c'est déplacé de: 5.385164807134504 m entre son emplacement précédent et son emplacement actuel

Process finished with exit code 0
```

Figure 54 : Sprint 4.2 – Résultat KNN console

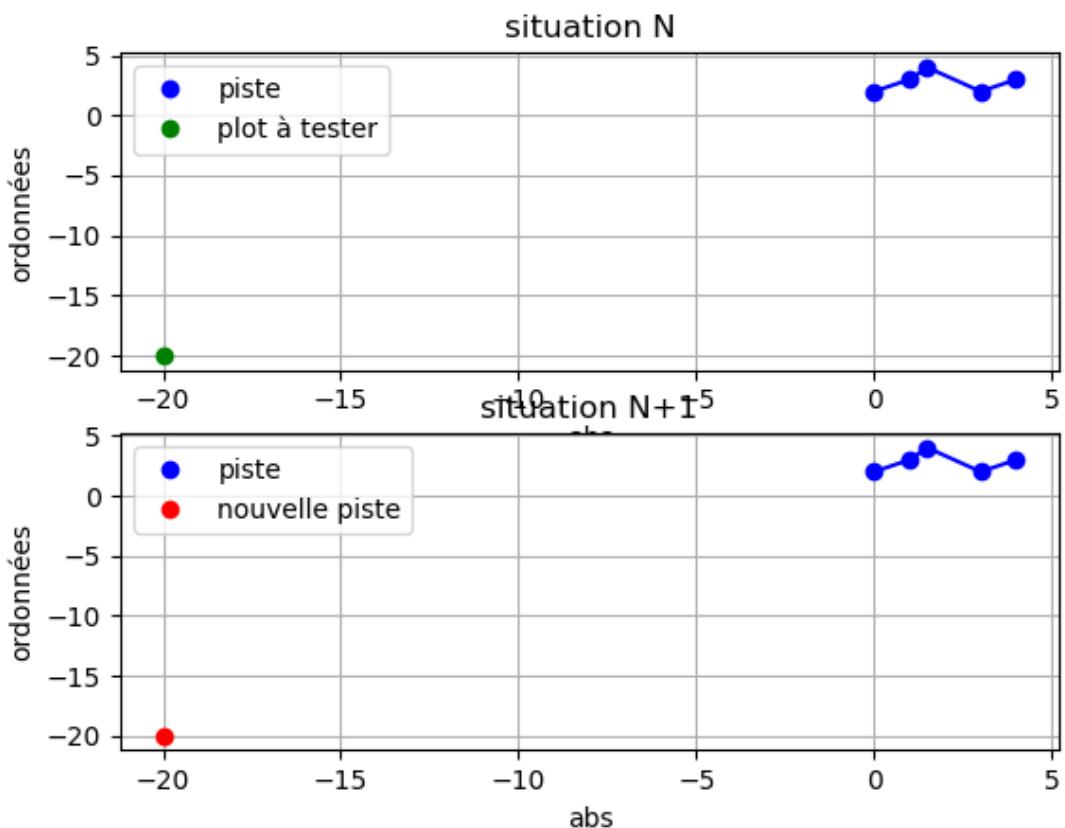


Figure 55 : Sprint 4.2 – Initialisation KNN graphique

```
C:\Users\emili\anaconda3\python.exe C:/Users/emili/OneDrive/Bureau/2A/0E
le plot n'appartient à aucune piste, on en crée une nouvelle
voisins: []
nouvelle affectation
le plot n'appartient pas à la piste
la nouvelle piste associée porte le numéro 34
nouvelle piste: [[-20, -20, 34]]
piste : [[0, 2, 3], [1, 3, 3], [1.5, 4, 3], [3, 2, 3], [4, 3, 3]]
le nouveau point est à : 28.284271247461902 m du radar

Process finished with exit code 0
```

Figure 56 : Sprint 4.2 – Initialisation KNN console

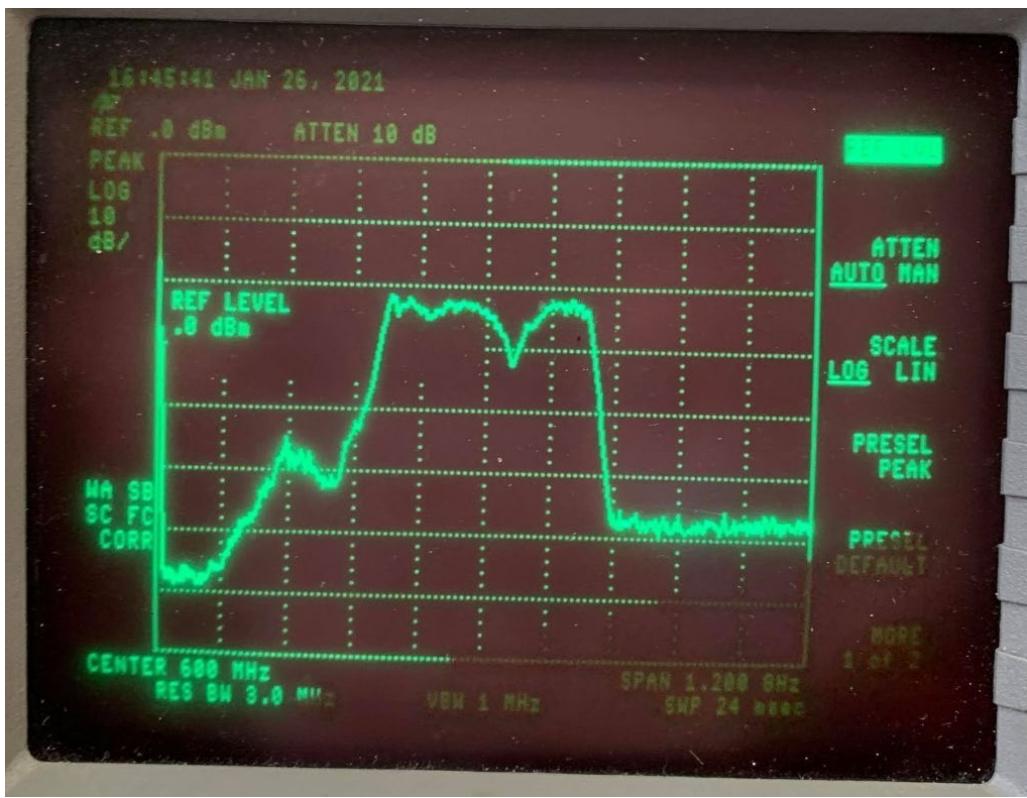


Figure 57 : Sprint 4.2 – Production d'un signal en intérieur

Sprint 4.3 : 23 Février au 2 mars

A faire	En cours	Fait
	Enrichissement de la bibliothèque de fonctions de calcul commencée au sprint précédent	Mise en place de la stratégie d'initialisation de piste et de suivi de 2 cibles pour l'algorithme de tracking des knn en 1D
	Mise en forme du signal acquis en array numpy pour clarifier, automatiser et améliorer (calcul Doppler) les calculs de post traitement (Jany et Julien)	
Envisager le développement d'une IHM	Mise en place de la fonction permettant de détruire les pistes non entretenues par l'algorithme de tracking knn	
	Manipulation : émettre en extérieur et pouvoir analyser les données	

Tableau 6 : Sprint 4.3

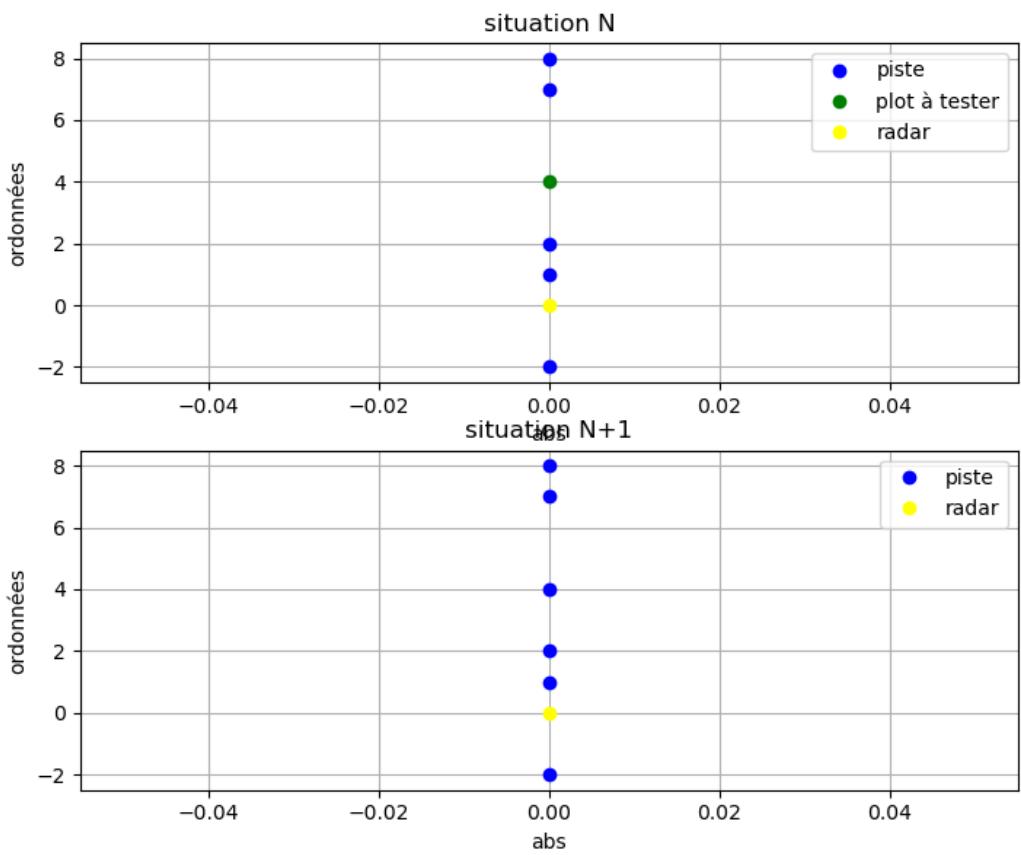


Figure 58 : Sprint 4.3 – KNN 1D avec 2 cibles

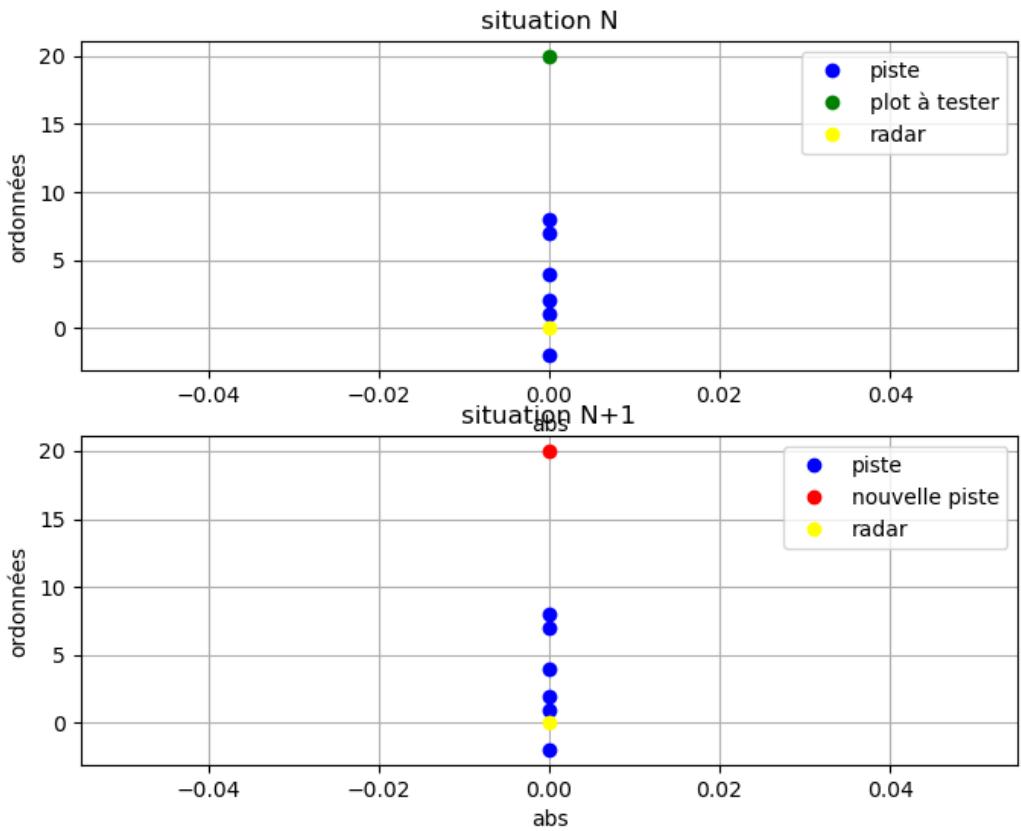


Figure 59 : Sprint 4.3 – KNN 1D suivi et mise à jour

```

C:\Users\emili\anaconda3\python.exe C:/Users/emili/OneDrive/Bureau/2A/OBShyper/traking/codes/knn_1D.py
let's start
voisins: [[8, 1], [7, 1], [2, 1]]
le plot appartient à la piste
piste mise à jour: [[7, 1], [8, 1], [1, 1], [-2, 1], [2, 1], [4, 1]]
le nouveau point est à : 4.0 m du radar
le point c'est déplacé de: 2.0 m entre son emplacement précédent et son emplacement actuel
piste 1 updated 1er tour [[7, 1], [8, 1], [1, 1], [-2, 1], [2, 1], [4, 1]]
une deuxième piste ? []
deuxième tour
le plot n'appartient à aucune piste, on en crée une nouvelle
voisins: []
nouvelle affectation
le plot n'appartient pas à la piste
la nouvelle piste associée porte le numéro 34
nouvelle piste: [[20, 34]]
piste : [[7, 1], [8, 1], [1, 1], [-2, 1], [2, 1], [4, 1]]
le nouveau point est à : 20.0 m du radar
piste 1 updated [[7, 1], [8, 1], [1, 1], [-2, 1], [2, 1], [4, 1]]
une deuxième piste ? [[20, 34]]

Process finished with exit code 0

```

Figure 60 : Sprint 4.3 – KNN 1D suivi et mise à jour

Sprint 4.3 : 9 mars au 23 mars

A faire	En cours	Fait
Début des manipulations et premières acquisitions avec le matériel en extérieur		Transformation en matrice pour le code avant CFAR
Tentatives d'application de la chaîne entière algorithmes à cette acquisition (toutes l'équipe)	Mise en place de la fonction permettant de détruire les pistes non entretenues par l'algorithme de tracking knn	
Tests unitaires avec différents types de signaux pour l'entrée de extérieur (photo 1) zoom (photo 1bis)	Première acquisition en	
Nécessité d'avoir nos propres enregistrements pour tester tous les algorithmes. En particulier le tracking		
	Mise en place du seuil pour l'algorithme de tracking avec les données et le contexte de l'acquisition via la fonction calcul_seuil	
Tests en extérieur: prendre en compte les difficultés liées à la fréquence du signal : beaucoup d'atténuation		Acquisition d'un faux mouvement de cible de 2m à 2m30 par tranches d'environ 2cm, puis de 5m à 5m30.
Développement d'une IHM		

simple		
	Agile : mise en forme des documents de sprints + récupérer des "preuves" (images, acquisitions...) pour le site Mahara	
28 mars : RAPPORT ECRIT		

Tableau 7 : Sprint 4.4

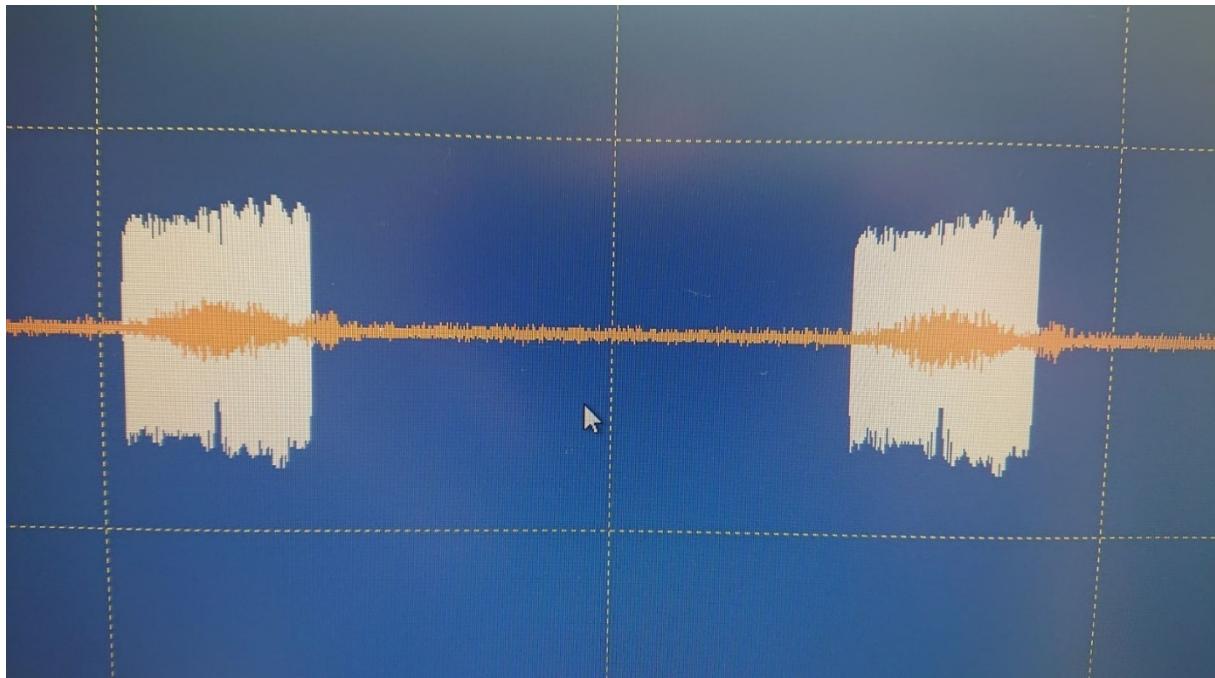


Figure 61 : Sprint 4.4 – Premières acquisitions en extérieur

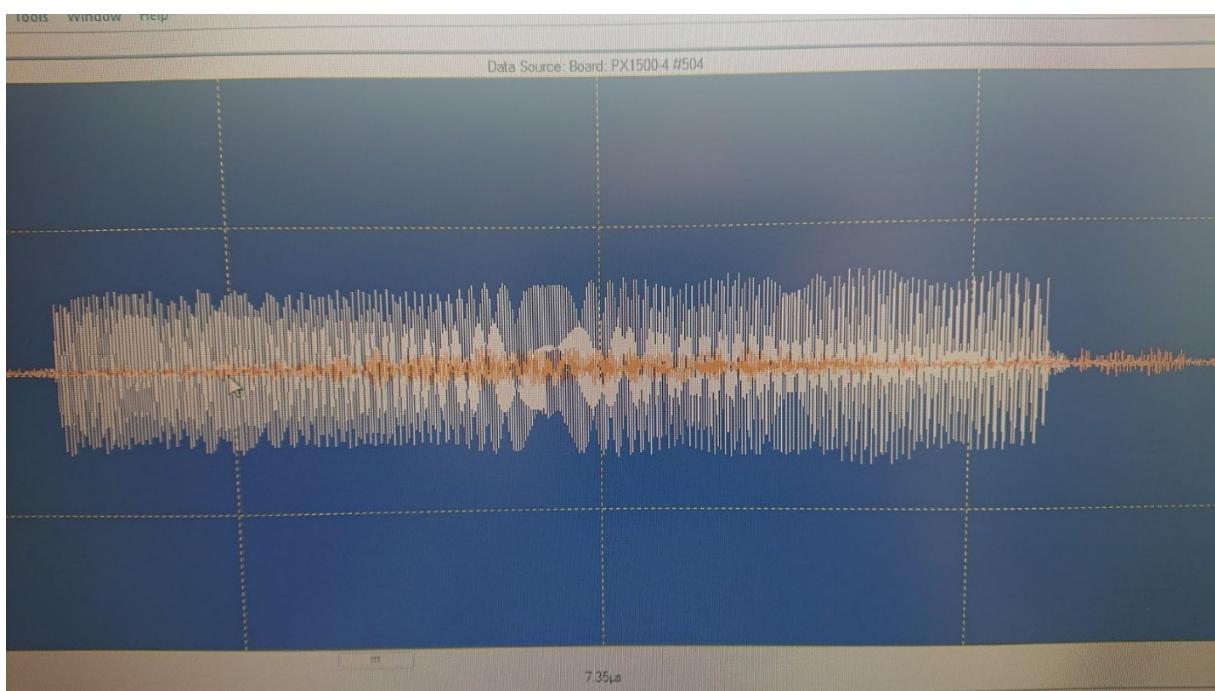


Figure 62 : Sprint 4.4 – Premières acquisitions en extérieur (zoom)

Conclusion

Au cours de ce premier semestre et lors de la phase de mise en place et de développement de notre projet nous avons pu travailler simultanément sur différents fronts. Tout d'abord nous nous sommes concentrés sur les principes physiques et la théorie soutenant le fonctionnement d'un radar hyperfréquences, cela nous a conduit naturellement à la réalisation d'une partie de ce dernier à l'aide de composants sur baie. Nous avons pu ainsi mettre en place une chaîne produisant une onde hyperfréquence prête à être communiquée à une antenne de radar. Nous avons aussi travaillé sur la programmation et le traitement du signal, partie essentielle pour exploiter les signaux communiqués par un radar. Nous avons donc implémenté différents codes avec des buts précis : récupérer un signal bruité pour le rendre exploitable, détecter dans ce signal des points trahissant la présence d'une cible et enfin exploiter ces points pour effectuer le suivi de ces cibles. Un à un, ces codes sont fonctionnels, nous cherchons donc maintenant à les faire travailler ensemble et, à terme, avec en entrées nos acquisitions faites sur le terrain. Nous avons aussi travaillé sur nos capacités relationnelles afin de relever les défis imposés par le confinement. Les leçons que nous en avons tiré notamment sur la communication et la gestion de la charge de travail nous seront utiles car les défis auxquels nous avons fait face seront renouvelés car la reprise des cours en présentiel est prévu en février. Nous avons pour la seconde partie du projet de nouveaux objectifs et perspectives. Nous voulons travailler avec le matériel et pouvoir à terme générer des ondes plus complexes dans le but d'avoir un radar fonctionnel capable de détecter un objet sur le terrain de sport de l'ENSTA Bretagne par exemple. Concernant la programmation, nous souhaitons améliorer nos algorithmes comme avancé précédemment mais aussi travailler en parallèle sur d'autres aspects comme le traitement de l'information en temps réel, la furtivité mais aussi, si cela est possible, l'implémentation d'une interface homme-machine.

Arrivés au terme de ce second semestre, marquant l'aboutissement de notre projet, nous pouvons dresser humblement, mais fièrement, un bilan honorable. Nous aurons finalement réussi à joindre tous les bouts de la chaîne commençant à la création d'un chirp à l'aide du module RS AFQ 100B, jusqu'à un algorithme capable de distinguer un objet mouvant du paysage radiofréquence de nos acquisitions, en extérieur de surcroit. Au-delà même de nos résultats purement scientifiques, notre équipe nous aura permis d'en apprendre un peu plus sur la gestion de projet d'un point de vue logistique, mais également humain. Enfin, comme nous l'exposions en préambule de ce rapport, les enseignements fournis par nos encadrants nous aurons permis de prendre de l'avance sur notre programme académique, avantage non négligeable au regard de la densité des disciplines abordées lors de cette première année de spécialisation. C'est pourquoi l'équipe souhaite conclure ces pages par nos remerciements, de bon cœur !

Bibliographie :

- [1] : Rapport de mi-parcours
https://moodle.ensta-bretagne.fr/pluginfile.php/45236/assignsubmission_file/submission_files/79233/Compte_rendu_avancement_projet.pdf
- [2] : <https://www.techno-science.net/definition/2408.html>
- [3] : <https://www.radartutorial.eu/08.transmitters/Compression%20d%E2%80%99impulsion.fr.html>
- [4] : https://fr.wikipedia.org/wiki/Compression_d%27impulsion
- [5] : Christian Wolff. (1998). Radartutorial. <https://www.radartutorial.eu/index.en.html>
- [6] : David Roche. (date non communiquée). Pixees.
https://pixees.fr/informatiquelycee/n_site/nsi_prem_knn.html
- [7] : Onel Harrison. (2018). Towards Data Science. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [8] : bar-shalom, Yaakov & Daum, Fred & Huang, Jim. (2010). The probabilistic data association filter. Control Systems, IEEE. 29. 82-100.10.1109/MCS.2009.934469.
https://www.researchgate.net/publication/224083228_The_probabilistic_data_association_filter
- [9] : Kirubarajan, Thia & bar-shalom, Yaakov. (2004). Probabilistic Data Association Techniques for Target Tracking in Clutter. Proceedings of the IEEE. 92. 536 - 557. 10.1109/JPROC.2003.823149.
https://www.researchgate.net/publication/2986277_Probabilistic_Data_Association_Techniques_for_Target_Tracking_in_Clutter
- [10] : Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In Proceedings of the IEEE international conference on computer vision (pp. 4696-4704)
https://www.cc.gatech.edu/~ckim314/papers/MHTR_ICCV2015.pdf
- [11] : Amditis, Angelos & Thomaidis, George & Maroudis, Pantelis & Lytrivis, Panagiotis & Karaseitanidis, Ioannis. (2012). Multiple Hypothesis Tracking Implementation. 10.5772/33583.
https://www.researchgate.net/publication/224829356_Multiple_Hypothesis_Tracking_Implementation

Table des figures :

- Figure 1 : Chaîne radar type - www.radartutorial.eu
- Figure 2 : RS AFQ 100B - www.rohde-schwarz.com
- Figure 3 : Configuration d'un chirp spécifique sur K6 Pulse Sequencer - P-O. Pedreno
- Figure 4 : RS SMBV 100A - www.rohde-schwarz.com
- Figure 5 : Convertisseur GeoSync Microwave - www.geosyncmicrowave.com
- Figure 6 : Montage type de notre système - J. Duchemann
- Figure 7 : Chaîne fonctionnelle de notre système - P-O. Pedreno
- Figure 8 : Signal témoin en sortie du splitter - P-O. Pedreno
- Figure 9 : Signal lu après réflexion sur un mur à 1 mètre - P-O. Pedreno
- Figure 10 : Montage réel du système ObsHyper (gauche) – cible (droite) - P-O. Pedreno
- Figures 11 et 12 : Positionnement angulaire correct du réflecteur (gauche) – amplitude réfléchie (droite) - P-O. Pedreno
- Figures 13 et 14 : Positionnement angulaire incorrect du réflecteur (gauche) – amplitude réfléchie (droite) - P-O. Pedreno
- Figures 15 à 17 : Acquisition continue, nombre de points et durée de l'acquisition - P-O. Pedreno
- Figure 18 : Signal obtenu en sortie d'algorithme – J. Duchemann
- Figure 19 : Spectre obtenu en sortie d'algorithme – J. Duchemann
- Figure 20 : Exemple de chirp linéaire – fr.wikipedia.org/wiki/Chirp
- Figure 21 : Signal émis en rouge et échos atténués en bleu - fr.wikipedia.org/wiki/Compression_d%27impulsion

Figure 22 : Signal émis en rouge et échos atténusés en bleu –
fr.wikipedia.org/wiki/Compression_d%27impulsion

Figure 23 : Exemple d'intercorrélation entre un signal émis et son retour différé et bruité – J. Duchemann

Figure 24 : Exemple d'intercorrélation entre une succession de signaux émis et leur retours différés et bruités – J. Duchemann

Figure 25 : Principe du “Cell-averaging CFAR” - www.radartutorial.eu/01.basics/Taux de fausses alarmes.fr.html

Figure 26 : Schéma de principe des cellules testées - fr.mathworks.com/help/phased/ug/constant-false-alarm-rate-cfar-detection.html

Figure 27 : Résultat de l'algorithme CFAR – J. Lahlouh

Figure 28 : Résultat de l'algorithme CFAR – J. Lahlouh

Figure 29 : Résultat de l'algorithme CFAR sur un signal type de nos appareils – J. Lahlouh

Figure 30 : Détection de cible après filtrage adapté sur un signal brut – J. Lahlouh

Figure 31 : Zone de validation autour de la position estimée – E. Simon

Figure 32 : Arbre formé par les différentes hypothèses nom original “Tree representation of the formed hypotheses” © 2012 DOI 10.5772/33583 [11]

Figure 33 : Jeu de données « propres » – E. Simon

Figure 34 : Un pas de l'algorithme de tracking en 2D. Affectation d'un plot à la piste adéquate avec 3 voisins – E. Simon

Figure 35 : Initialisation d'une nouvelle piste avec un plot délibérément placé au-dessus du seuil de l'algorithme de tracking 2D – E. Simon

Figure 36 : Mise à jour de la piste avec l'algorithme de tracking en 1D avec 3 voisins. Points de la piste initialisés au préalable et plot avec une coordonnée cohérente. (Test unitaire) – E. Simon

Figure 37 : Initialisation d'une nouvelle piste avec 3 voisins avec un plot délibérément placé loin du radar. Les 2 pistes présentes ont été initialisées avec un jeu de données "propres" (test unitaire) – E. Simon

Figure 38 : Algorithme en sortie du CFAR, sur acquisition – E. Simon

Figure 39 : Initialisation avec les données du CFAR – E. Simon

Figure 40 : Algorithme en sortie du CFAR, sur acquisition – E. Simon

Figure 41 : Continuation de l'algorithme de suivi – E. Simon

Figure 42 : Algorithme en sortie du CFAR, sur acquisition – E. Simon

Figure 43 : Fin de l'algorithme – E. Simon

Figure 44 : Piste finale – E. Simon

Figure 45 : cycle en V- Max Godenn dans Cycle en V sur Planilog

<https://www.planilog.com/fr/glossaire/cycle-en-v>

Figure 46 : Chaîne mise en place pour générer le signal – E. Simon

Figures 47 et 48 : Diagramme de Gantt en début de projet – P-O. Pedreno

Figures 49 et 50 : Diagramme de Gantt aux vacances de Noël – P-O. Pedreno

Figures 51 et 52 : Diagramme de Gantt du 4e semestre – E. Simon

Figure 53 : Sprint 4.2 – Résultat KNN graphique – E. Simon

Figure 54 : Sprint 4.2 – Résultat KNN console – E. Simon

Figure 55 : Sprint 4.2 – Initialisation KNN graphique – E. Simon

Figure 56 : Sprint 4.2 – Initialisation KNN console – E. Simon

Figure 57 : Sprint 4.2 – Production d'un signal en intérieur – E. Simon

Figure 58 : Sprint 4.3 – KNN 1D avec 2 cibles – E. Simon

Figure 59 : Sprint 4.3 – KNN 1D suivi et mise à jour – E. Simon

Figure 60 : Sprint 4.3 – KNN 1D suivi et mise à jour – E. Simon

Figure 61 : Sprint 4.4 – Premières acquisitions en extérieur – P-O. Pedreno

Figure 62 : Sprint 4.4 – Premières acquisitions en extérieur (zoom) – P-O. Pedreno

Table des tableaux :

Tableau 1 : Tableau des fonctions du système – E. Simon

Tableau 2 : Tableau des composants

Tableau 3 : Liste des tests unitaires effectués sur les algorithmes

Tableau 4 : Sprint 4.1 – E. Simon

Tableau 5 : Sprint 4.2 – E. Simon

Tableau 6 : Sprint 4.3 – E. Simon

Tableau 7 : Sprint 4.4 – E. Simon

Annexes :

Algorithme de traitement du signal : <https://pastebin.com/GMzSXJtf>

Algorithme CFAR : <https://pastebin.com/FcR4fuQV>

Algorithme K-NN : <https://pastebin.com/J3UcCGg3>

Filtre de Kalman : <https://pastebin.com/LSWZcFM6>

Tous les algorithmes (et la chaîne principale) sont disponibles sur le github du projet à l'adresse :

<https://github.com/AspirantMoutarde/radar-signal-post-processing?fbclid=IwAR373RMK-8Y0YcFCaU9-rXYqGL4b5X6kh8Yhl5k2FgGm5RB3jt1xxCPbbp0>