Utilisation de méthodes d'apprentissage automatique pour des tâches de pistage radar et de classification d'aéronefs

Sami Jouaber

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES PARIS

Utilisation de méthodes d'apprentissage automatique pour des tâches de pistage radar et de classification d'aéronefs

Soutenue par
Sami JOUABER
Le 20/09/2022

École doctorale n°621
ISMME

Spécialité
Informatique temps réel, robotique et automatique

Composition du jury :

Audrey GIREMUS Professeure, Université de Bordeaux, IMS	*Rapporteure & Présidente*
Frédéric BOUCHARA Maître de conférences, Université de Toulon, LIS	*Rapporteur*
François SEPTIER Professeur, Université Bretagne Sud, LMBA	*Examinateur*
Silvère BONNABEL Professeur, Université de la Nouvelle-Calédonie, ISEA	*Directeur de thèse*
Santiago VELASCO-FORERO Chargé de recherche, Mines ParisTech, CMM	*Examinateur*
Marion PILTÉ Docteure, THALES LAS	*Examinatrice*

MINES PARIS | PSL

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Généralités sur le traitement radar

1.1.1 Historique

Le mot radar est un acronyme pour *Radio Detection and Ranging*. Pour une introduction au radar, le lecteur pourra consulter DARRICAU [2002] ou BLANCHARD [2004]. Le terme est apparu en 1940 dans l'armée américaine, et était alors un nom de code. Cela dit, les radars proprement dits étaient apparus bien avant. Les premiers systèmes radar avaient été développés au début du vingtième siècle pour l'évitement de collision pendant des tâches de navigation. Les premières expériences ayant été conduites par Nikola Tesla en 1900 et Christian Hülsmeyer en 1904. Les grandes avancées sont cela étant dues à la course aux armes des années trente et à la seconde guerre mondiale, pendant laquelle plusieurs grands pays ont simultanément développé leurs systèmes radar.

Les radars "modernes" datent de la seconde moitié du vingtième siècle. Il s'agit de radars à compression d'impulsions. Leur développement a conduit aux radars modernes, comme les radars à antenne active. Il existe à ce jour d'autres types de radar, qu'ils soient bi-statiques ou mono-statiques, actifs ou passifs, primaires ou secondaires. Ces radars ont des utilisations différentes, qu'elles soient civiles ou militaires.

La théorie du radar est à la croisée de plusieurs disciplines scientifiques. La théorie des antennes, le traitement du signal, les aspects numériques et le traitement algorithmique, ainsi que le filtrage statistique sont les principaux domaines scientifiques impliqués. La présente thèse porte essentiellement sur le filtrage statistique du signal pour le pistage, et cherche à explorer de nouvelles voies sur l'utilisation de techniques modernes d'intelligence artificielle pour rendre les filtres plus adaptatifs.

1.1.2 Description générale des systèmes radar

Le lecteur pourra consulter les ouvrages DARRICAU [2002] ou SKOLNIK [1970] pour plus de détails. Nous nous contenterons ici d'une description rapide. Un radar émet une onde électromagnétique, qui est réfléchie par les objets (une cible), voir figure 1.1. Il s'agit alors d'analyser ce signal réfléchi, au travers d'un récepteur.

La conception d'antennes est difficile, puisque les signaux réfléchis sont beaucoup plus faibles que les signaux émis, et corrompus par des "bruits" divers.

Le radar primaire mesure la position de la cible sans que cette dernière ne coopère. La position est généralement donnée au travers d'une mesure de distance à la cible, complétée par l'azimuth et l'élévation de la cible, i.e., des angles. Ce système de coordonnées impliqué dans la mesure radar est représenté sur la figure 1.2.

La distance est mesurée au travers du temps mis par l'onde pour aller jus-

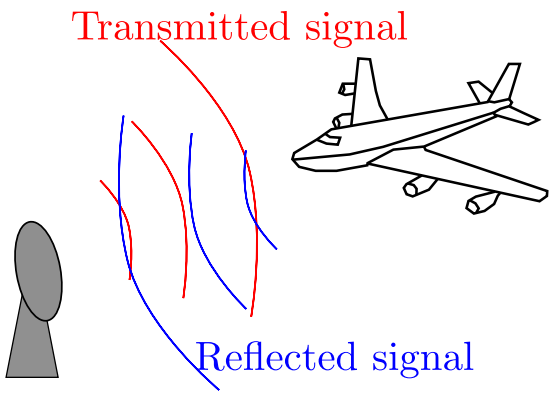


FIGURE 1.1 – Principe du radar

qu'à la cible et revenir. Ce temps est multiplié par la célérité de l'onde émise, puis divisé par deux, pour retrouver la distance. La mesure des angles d'azimuth et d'élévation est rendue possible par le fait que les faisceaux radars sont directionnels. Cela étant, cette mesure d'angles est généralement moins précise que la mesure de distance qui est extrêmement précise dans les radars modernes.

Le radar secondaire ne mesure pas directement la position de la cible. Il permet d'identifier la présence d'une cible, et requiert qu'elle l'informe de sa propre position. Ce genre de radar peut être utile dans un contexte civil (surveillance du trafic aérien) mais inutile dans un contexte militaire. Les radars secondaires sont également à même de recevoir l'information dite ADS-B (Automatic Dependent Surveillance - Broadcast) émise par les avions et contenant des éléments de position, type d'appareil etc. Dans cette thèse, nous utiliserons les données ADS-B pour entraîner des algorithmes à reconnaître automatiquement une cible non-coopérative, qui refuserait de communiquer ces éléments, sur la base de sa trajectoire.

1.1.3 Traitement numérique

L'unité en charge du traitement numérique reçoit l'information brute du récepteur. Elle doit alors créer les "plots", qui sont censés représenter les objets présents dans l'image radar, ainsi que des informations diverses sur ces objets. Le test de détection est un test de type statistique, destiné à maximiser la probabilité de détection tout en se maintenant en dessous d'un seuil prédéfini de fausses alarmes.

Le signal radar, basé sur des mesures d'une onde électromagnétique en temps contenu, est numérisé, et plusieurs traitements lui sont alors appliqués.

1.1.4 Pistage

Un radar assure la détection des objets et leur pistage. Le but du pistage est, une fois l'objet labellisé, et suivi, d'inférer ses paramètres cinématiques et d'af-

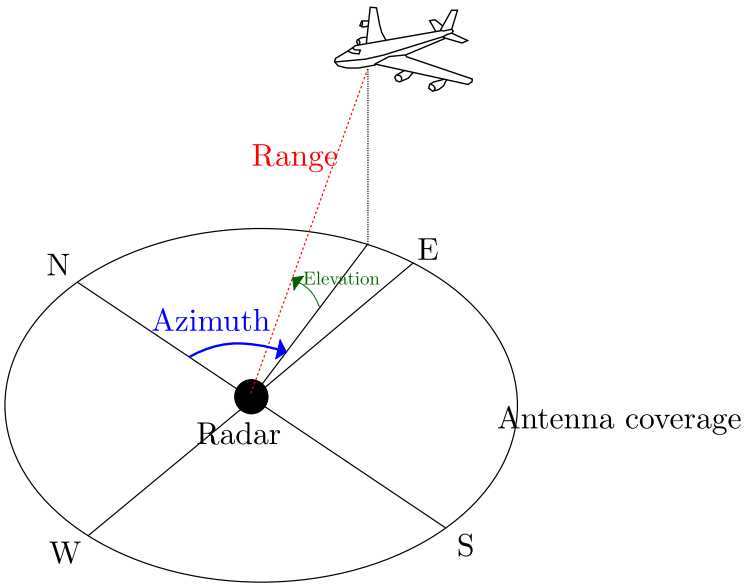


FIGURE 1.2 – Mesures radar

finer sa localisation, i.e., la mesure brute de sa distance et des angles associés.

Une piste correspond à l'évolution d'une cible détectée par le système. Les observations sont entachées de plusieurs types d'incertitudes. Tout d'abord, les fausses alarmes, ensuite, les incertitudes associées à la mesure radar même, le capteur n'étant jamais parfait, voir par exemple CURRY [2005]. Même si les sources d'erreur sont multiples, celle qui prédomine est une erreur dépendant du rapport signal sur bruit, qui tend à dégrader la mesure. Le but du pistage est d'analyser les plots pour en fabriquer des pistes. Une piste contient le label de l'objet, des descripteurs cinématiques (position, vitesse) et une possible identification de l'objet ainsi que d'autres informations utiles à l'opérateur.

Afin de pister un objet, il faut d'abord initialiser une piste. Il faut ensuite associer les plots à une piste existante, en évitant les fausses alarmes. Le filtrage est la partie algorithmique qui infère des données cinématiques de la cible et réduit (filtre) le bruit de mesure de position (voire prédit son évolution) à partir des données piste. Finalement, il existe un algorithme de plus haut niveau qui ordonne les tâches, ou décide de supprimer une piste lorsqu'elle est considérée comme perdue.

1.2 Algorithmes de pistage

Dans cette thèse, nous nous concentrons plus spécifiquement sur la tâche de pistage. En radar, la tâche de pistage consiste, en substance, à inférer en temps réel la vitesse d'une cible, à affiner la connaissance de sa position, ainsi qu'à être capable de prédire à court terme son évolution, à la lumière de me-

sures de sa position, que l'on va supposer comme correspondant bien à la cible en question, mais entachées d'erreurs de mesures (les bruits).

La difficulté du pistage radar est alors que le modèle cinématique de la cible est inconnu, ainsi l'incertitude ne découle pas seulement des bruits de mesures aléatoires, mais aussi du mouvement inconnu de la cible. La tâche de conception d'algorithmes de pistage demande alors d'une part d'élaborer un modèle cinématique statistique pour la cible (ou une banque de modèles), et d'autre part de régler les paramètres de bruits de mesure. Cette dernière tâche repose sur les caractéristiques des capteurs de mesure radar. En revanche, la première tâche offre au concepteur une grande latitude : il faut non-seulement définir un ou des modèles cinématiques plausibles, mais aussi et conjointement associer un système de lois statistiques à ces modèles puisque la cible ne les suivra jamais parfaitement.

Nous donnons maintenant un exposé mathématique des filtres principaux utilisés en radar. Nous commencerons par une présentation détaillée du filtre de Kalman, car il joue un rôle central dans cette thèse, et également parce qu'il permet de bien comprendre ce que fait un filtre statistique. A cet égard sa présentation détaillée est pédagogique.

1.2.1 Le filtre de Kalman

Equations du filtre Considérons un système dynamique discret évoluant selon l'équation

$$X_{t+1} = AX_t + W_t \tag{1.1}$$

où $X_t \in \mathbb{R}^n$ est une variable aléatoire vectorielle avec $X_0 \sim \mathcal{N}(\mu_0, P_0)$, et $W_t \sim \mathcal{N}(0, Q)$ une variable gaussienne indépendante de toutes les autres variables en jeu. On voit par les propriétés des vecteurs gaussiens que pour tout temps $t \in \mathbb{N}$ la variable X_t est gaussienne. Supposons qu'on mesure de façon bruitée une partie des coordonnées de l'état, *i.e.* on mesure $Y_t = HX_t + V_t$ avec $V_t \sim \mathcal{N}(0, R)$ une variable gaussienne indépendante de toutes les autres variables en jeu.

La situation est la suivante : les variables X_t et Y_t sont des variables aléatoires, du fait de l'incertitude initiale sur l'état (encodée dans la variable X_0) et du caractère aléatoire des bruits. *Mais*, X_t et Y_t sont liées, puisque Y_t est une fonction de X_t (à un bruit près). L'utilisateur mesure alors des valeurs de la sortie Y, à savoir les réels $y_0, \cdots y_t$. Que sait-on sur le système ? On sait la chose suivante (et on ne sait *que* cela) : l'état du monde $\omega \in \Omega$ qui sous-tend X_0 et la séquence des bruits $W_0, W_1, \cdots$ et $V_0, V_1, \cdots$, est tel que

$$Y_0(\omega) = y_0, \cdots, Y_t(\omega) = y_t$$

Alors, comment estimer au mieux l'état courant $X_t(\omega)$? Cela reste une variable aléatoire, on ne pourra pas trouver sa valeur exacte. Mais on a maintenant de

l'information sur cette variable à travers l'information apportée par les valeurs mesurées $Y_0(\omega), \cdots$.

Plus précisément, le but du filtrage de Kalman est de trouver la meilleure estimée, au sens des moindres carrés, de l'état X_t au vu de toutes les mesures passées $y_0, \cdots y_t$, valeurs mesurées de $Y_0, Y_1, \cdots, Y_t$.

Estimée du filtre de Kalman $\hat{X}_t$ = meilleure estimation de X_t au vu des mesures passées

Mathématiquement, cela implique

Estimée du filtre de Kalman recherchée définie par $\hat{X}_t = E(X_t \mid y_0, \cdots y_t)$

Les équations reposent sur plusieurs étapes récursives. Supposons que l'on ait calculé au temps t la quantité $\hat{X}_t = E(X_t \mid y_0, \cdots y_t)$ et que ce soit une variable gaussienne de matrice de covariance P_t. On calcule $\hat{X}_{t+1}$ de la façon suivante.

— **Prédiction** : la meilleure estimation de X_{t+1} au vu des mesures $y_0, \cdots y_t$ est

$$\hat{X}_{t+1|t} = A\hat{X}_t$$

et sa matrice de covariance vaut $P_{t+1|t} = AP_tA^T + Q$.

— Calcul de l'**innovation** : pour tenir compte de la mesure au temps $t+1$ on calcule la quantité

$$z_{t+1} = y_{t+1} - H\hat{X}_{t+1|t}$$

appelée innovation. On calcule aussi la **matrice de gain** du filtre de Kalman

$$K_{t+1} = P_{t+1|t}H^T(HP_{t+1|t}H^T + R)^{-1}$$

— **Mise à jour** : la meilleure prédiction tenant compte de la mesure y_{t+1} s'écrit

$$\hat{X}_{t+1} = \hat{X}_{t+1|t} + K_{t+1}z_{t+1}$$

et sa matrice de covariance vaut $P_{t+1} = (I - K_{t+1}H)P_{t+1|t}$ où I désigne la matrice identité de dimension $n \times n$.

On notera que le filtre de Kalman ne se contente pas de donner la meilleure prédiction $\hat{X}_t$ de l'état du système au vu des observations passées, il donne aussi sa matrice de covariance P_t. Cela permet notamment d'attribuer une confiance à la mesure, ou de trouver des ellipsoïdes de confiance où l'état X_t a de fortes probabilités de se trouver.

Le filtre de Kalman calcule ainsi récursivement (et donc en temps réel) la densité conditionnelle, totalement connue à travers ses deux premiers moments μ_t, P_t. On a en effet :

La densité conditionnelle $p(x_t \mid y_0, \cdots y_t)$ est la densité d'une loi gaussienne $\mathcal{N}(\mu_t, P_t)$

Démonstration des équations On va raisonner par récurrence. Supposons que l'on a montré que la densité de X_t sachant $y_0, \cdots y_t$, notée $p(x_t \mid y_0, \cdots y_t)$, est une gaussienne de paramètres $\mathcal{N}(\mu_t, P_t)$. Nous avons les points suivants

— X_{t+1} conditionné par $y_0, \cdots y_t$ est une gaussienne de moyenne et covariance

$$\mu_{t+1|t} = E(Ax_t + W_t | y_0, \cdots, y_t) = A\mu_t, \quad P_{t+1|t} = AP_tA^T + Q$$

En effet, la première équation découle de la linéarité de l'espérance conditionnelle, et la seconde d'un simple calcul de covariance et du fait que les covariances de vecteurs indépendants s'ajoutent.

— de même Y_{t+1} conditionné par $y_0, \cdots y_t$ est une gaussienne de moyenne et covariance

$$\nu_{t+1|t} = H\mu_{t+1|t}, \quad HP_{t+1|t}H^T + R$$

et $E[(Y_{t+1} - \nu_{t+1|t})(X_{t+1} - \mu_{t+1|t})^T] = HP_{t+1|t}$ car $Y_{t+1} = HX_{t+1} + V_{t+1}$, mais V_{t+1} est indépendant du reste et de moyenne nulle donc $E(V_{t+1}(X_{t+1} - \mu_{t+1|t})^T) = E(V_{t+1})E(X_{t+1} - \mu_{t+1|t})^T = 0$.

— on déduit du point précédent que la variable $(X_{t+1}, Y_{t+1})^T$ conditionnée par $y_0, \cdots y_t$ est une gaussienne de moyenne et de matrice de covariance

$$\begin{pmatrix} \mu_{t+1|t} \\ \nu_{t+1|t} \end{pmatrix}, \quad \begin{pmatrix} P_{t+1|t} & P_{t+1|t}H^T \\ HP_{t+1|t} & HP_{t+1|t}H^T + R \end{pmatrix}$$

Une simple application de résultats classiques sur le conditionnement des vecteurs gaussiens permet d'obtenir les équations de Kalman puisqu'on a conditionnellement aux valeurs mesurées $y_0, \cdots y_{t+1}$ que X_{t+1} est une gaussienne de moyenne

$$\mu_{t+1} = \mu_{t+1|t} + P_{t+1|t}H^T(HP_{t+1|t}H^T + R)^{-1}(y_{t+1} - \nu_{t+1|t})$$

et de covariance

$$P_{t+1} = P_{t+1|t} - P_{t+1|t}H^T(HP_{t+1|t}H^T + R)^{-1}HP_{t+1|t}$$

i.e. que la densité conditionnelle $p(x_{t+1} \mid y_0, \cdots y_{t+1})$ suit une loi gaussienne de paramètres $\mathcal{N}(\mu_{t+1}, P_{t+1})$.

1.2.2 Application au pistage et réglage

Lorsque l'on piste une cible, on peut supposer que l'état X_t regroupe les variables qui nous intéressent, à savoir dans un premier temps la position et la vitesse de la cible. Les mesures de position, si elles sont supposées transformées par un changement de coordonnées afin d'être exprimées dans un repère absolu (ce qu'on supposera pour simplifier ici) sont alors bien de la forme $Y_t = HX_t + V_t$ où la matrice H sélectionne les trois variables de position contenues dans X_t,

et où V_t représente un bruit de mesure aléatoire, i.e. une variable d'erreur de mesure dont on ne connaît pas la valeur particulière, mais dont on connaît le comportement statistique (sur un grand nombre de mesures) encodé au travers du paramètre matriciel R, essentiellement car on peut le relier aux performances du "capteur" radar.

On voit qu'en pistage, la difficulté se concentre autour du choix des paramètres A_t, Q_t du modèle cinématique statistique (1.1). Un premier modèle répandu dans l'industrie et bien adapté aux cibles de type avions de ligne consiste à supposer que la vitesse est constante (plus précisément le vecteur vitesse est constant au cours du temps). Cela spécifie entièrement la matrice A_t. Bien sûr, ce n'est vrai qu'approximativement, et le vecteur aléatoire W_t encode un écart entre l'évolution de l'état vrai et celle selon le modèle ligne droite vitesse constante. Lors des lignes droites cet écart est potentiellement faible, mais il peut être important lors des manoeuvres. Le paramètre Q, qui représente l'amplitude de la dispersion entre évolution vraie et modèle, convient d'être réglé. On l'appelle *bruit de modèle*. Si les mouvements sont presque tout le temps des lignes droites, Q peut être de faible amplitude, mais pour une cible très manœuvrante il convient de le régler pour qu'il reflète une forte dispersion (voire de changer de modèle).

La présente thèse va se concentrer sur le problème du réglage de la matrice Q, et comment l'apprentissage automatique sur un grand nombre de trajectoires simulées peut nous fournir des méthodes de réglage de Q adaptatives, c'est-à-dire des méthodes à même de détecter automatiquement des manoeuvres et d'alors augmenter le paramètre Q dynamiquement, comme de le diminuer lors de mouvements rectilignes uniformes par exemple.

1.2.3 Le filtre de Kalman étendu

Lorsqu'on utilise un modèle non-linéaire pour la cible, i.e. qu'on suppose que la dynamique des variables d'état qui la caractérisent est de la forme

$$X_{t+1} = f(X_t, W_t) \tag{1.2}$$

avec un f une fonction non-linéaire, à la place du modèle (1.1) ; et/ou que la mesure de l'état s'écrit

$$Y_t = h(X_t) + V_t \tag{1.3}$$

avec une h une application non-linéaire, on ne peut plus appliquer le filtre de Kalman classique comme présenté ci-dessus. Un modèle non-linéaire peut être utilisé par exemple pour caractériser le mouvement d' une cible qui suivrait un mouvement balistique. En ce qui concerne la mesure, les modèles non-linéaires du type (1.3) prédominent en radar. En effet, l'état X_t contient la position de la cible en coordonnées absolues dans un repère fixe, alors que la mesure radar de la position s'obtient sous forme d'une distance et d'angles (coordonnées

sphériques). Le changement de coordonnées entre la position en coordonnées cartésienne (qui représente 3 composantes dans le vecteur d'état X_t) et les coordonnées dans lesquelles la mesure Y_t est obtenue est non-linéaire.

Dès les années soixante, lors des premières implémentations du filtre de Kalman dans le programme Apollo, les ingénieurs de la NASA ont trouvé une méthode conjointement avec Rudoph Kalman pour utiliser le filtre en présence de non-linéarités. L'idée est à chaque étape d'approximer les fonctions non-linéaires par des fonctions linéaires. Aussi, on écrit à l'étape de propagation on propage la moyenne $\hat{X}_t$ au travers du modèle sans bruit, i.e

$$\hat{X}_{t+1} = f(\hat{X}_t, 0).$$

En supposant l'incertitude concentrée autour de sa moyenne $\hat{X}_t$ et l'amplitude du bruit de modèle W_t assez réduite :

$$X_{t+1} = f(X_t, W_t) = f(\hat{X}_t + (X_t - \hat{X}_t), W_t) \approx f(\hat{X}_t, W_t) + A_t(X_t - \hat{X}_t) + G_t W_t$$

où A_t et G_t sont les matrices jacobiennes de f par rapport à X et W au point $(\hat{X}_t, 0)$. On est ainsi ramené à un modèle linéaire de type (1.1) pour l'erreur $(X_t - \hat{X}_t)$. On peut alors propager ses paramètres au travers des équations de Kalman, i.e. on a avant la mesure $(X_{t+1} - \hat{X}_{t+1}) \sim \mathcal{N}(0, A_t P_t A_t^T + G_t Q_t G_t^T)$.

Au moment de la mesure, on peut aussi écrire

$$Y_{t+1} - h(\hat{X}_{t+1}) = h(X_{t+1}) - h(\hat{X}_{t+1}) + V_t \approx H_{t+1}(X_{t+1} - \hat{X}_{t+1}) + V_t$$

avec H_{t+1} la matrice jacobienne de h au point $\hat{X}_{t+1}$, et on voit qu'on obtient alors approximativement une mesure linéaire de l'erreur $(X_{t+1} - \hat{X}_{t+1})$. Les équations de Kalman permettent de mettre à jour notre loi pour cette erreur, conditionnellement aux observations passées, jusqu'à Y_{t+1} incluse. On peut ensuite en déduire un approximation de la loi de X_t conditionnellement aux mesures.

Le filtre de Kalman étendu (EKF en anglais) consiste donc à se placer en écart à la trajectoire estimée $\hat{X}_t$, en supposant ces écarts petits, et à linéariser le modèle. Cela donne une approximation de ce que l'on cherche, mais sans garantie d'exactitude. Le filtre EKF fonctionne généralement très bien en pratique, mais il arrive qu'il soit mis en défaut sur des exemples particuliers.

1.2.4 Le filtre IMM

Le principe du filtre IMM (*Interacting Multiple Models*) proposé par BLOM et BAR-SHALOM [1988] et représenté en figure 1.3 est faire fonctionner plusieurs filtres de Kalman possédant des modèles d'évolution différent ou juste des bruits de modèles différents. Ces modèles sont mis en émulation et la confiance accordée à chacun d'entre eux au temps t_n est exprimée par le vecteur de probabilités p_n. Au début de chaque cycle, ces probabilités sont mises à jour à l'aide de la matrice de transition T à paramétrer (équation 1.4). Ensuite, une phase d'interaction permet de mélanger les états des différents modèles pour les rapprocher de

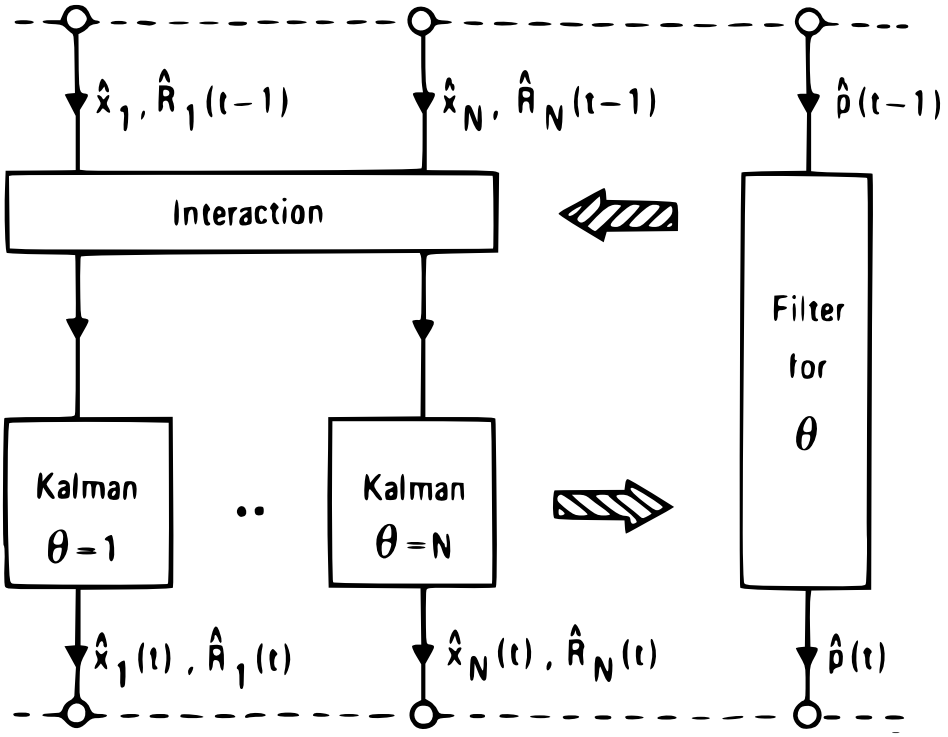


Fig. 1. The IMM algorithm.

FIGURE 1.3 – Schéma de fonctionnement du filtre IMM, issu de BLOM et BAR-SHALOM [1988].

ceux jugés plus fiables selon ces probabilités (équations 1.5 et 1.6). Après cela, les filtres effectuent leurs étapes de prédiction et d'estimation, puis les probabilités sont à nouveau mises à jour, cette fois-ci à l'aide des vraisemblances des estimations des filtres (équations 1.7 et 1.8. Enfin, le filtre exprime un état moyen pondéré par les probabilités (équations 1.9 et 1.10).

$$\tilde{p}_n = \mathrm{T}.p_{n-1} \tag{1.4}$$

$$(\tilde{\mathrm{X}}_{n-1})_m = \frac{1}{(\tilde{p}_n)_m} \sum_i (\mathrm{T}_n)_{mi} (p_{n-1})_i (\mathrm{X}_{n-1})_i \tag{1.5}$$

$$\begin{aligned}(\tilde{\mathrm{P}}_{n-1})_m = \frac{1}{(\tilde{p}_n)_m} (\sum_i (\mathrm{T}_n)_{mi} (p_{n-1})_i (\mathrm{P}_{n-1})_i \\ + (\mathrm{X}_{n-1} - \tilde{\mathrm{X}}_{n-1})_m . (\mathrm{X}_{n-1} - \tilde{\mathrm{X}}_{n-1})_m^{\mathrm{T}})\end{aligned} \tag{1.6}$$

$$(l_n)_m \propto |(\mathrm{S}_n)_m|^{-\frac{1}{2}} e^{-\frac{1}{2}\|(\mathrm{Y}_n)_m\|^2_{(\mathrm{S}_n)_m}} \tag{1.7}$$

$$(p_n)_m = \frac{l_m (\tilde{p}_n)_m}{\sum_i l_i (\tilde{p}_n)_i} \tag{1.8}$$

$$\mathrm{X}_n = \sum_m (p_n)_m (\mathrm{X}_n)_m \tag{1.9}$$

$$\mathrm{P}_n = \sum_m (p_n)_m ((\mathrm{P}_n)_m + ((\mathrm{X}_n)_m - \mathrm{X}_n).((\mathrm{X}_n)_m - \mathrm{X}_n)^{\mathrm{T}}) \tag{1.10}$$

1.3 Apprentissage automatique

1.3.1 Principe général de l'apprentissage

Le principe de l'apprentissage artificiel (machine learning en anglais) est de construire une application mathématique Φ définie par un nombre N de paramètres θ que l'on ajustera pour faire faire à la fonction la tâche voulue.

$$\begin{array}{rccc} \Phi\colon & \mathbb{R}^{\mathrm{N}} \times \mathrm{A} & \rightarrow & \mathrm{B} \\ & \theta, x & \mapsto & y \end{array} \tag{1.11}$$

Dans le cas d'un apprentissage supervisé, cet ajustement se fera grâce à une base d'exemples prenant la forme d'un ensemble de données d'entrée X pour lesquelles les sorties attendues Y pour notre application Φ sont connues. Pour mesurer l'erreur de Φ, on se dote d'une fonction de coût $\mathscr{H}$.

$$\begin{array}{rccc} \mathscr{H}\colon & \mathrm{B} \times \mathrm{B} & \rightarrow & \mathbb{R} \\ & y_1, y_2 & \mapsto & \mathscr{H}(y_1, y_2) \end{array} \tag{1.12}$$

L'entraînement consiste alors à optimiser les paramètres θ afin de minimiser l'erreur.

$$\theta^{opt} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathscr{H}(\Phi(\theta, \mathrm{X}_i), \mathrm{Y}_i) \tag{1.13}$$

Si nous prenons l'exemple très simple de la régression linéaire, Φ est une fonction affine et les paramètres θ sont le coefficient a et le biais b qui la définissent ($\Phi((a,b),x) = ax + b$). On optimise généralement la régression par les moindres carrés ($\mathscr{H}(y_1, y_2) = (y_1 - y_2)^2$). La solution analytique au problème d'optimisation est alors facile à trouver en cherchant les conditions d'annulation des dérivées partielles de $\mathscr{H}$ par a et b, ce qui revient à résoudre un système linéaire d'équations. Cependant, dans le cas général il n'est pas possible d'obtenir une solution analytique, et la recherche d'une solution au problème d'optimisation passe par un algorithme de descente de gradient. L'application qui à θ associe la valeur de la fonction de coût pour un jeu d'exemples fixé n'est en général pas convexe, la convergence de tels algorithmes vers le minimum global n'est pas garantie par la théorie. Cependant, dans la pratique, les algorithmes de descente de gradient se sont avérés plutôt efficaces pour résoudre les problèmes d'optimisation de l'apprentissage automatique moderne, souvent composés de couches de neurones sur lesquels nous reviendront.

De nombreuses variantes de la descente de gradient existent. La plupart sont stochastiques, c'est-à-dire que seule une partie des paramètres choisie aléatoirement est mise à jour. Elles ont parfois une inertie, c'est-à-dire que l'algorithme

pondère le gradient calculé à une étape avec ceux calculés aux étapes précédentes pour mettre à jour les paramètres. Parmi les plus connues on peut citer RMSProp [1], Adadelta par ZEILER [2012] ou Adam par KINGMA et BA [2014].

1.3.2 Réseaux de neurones

L'un des outils principaux utilisés pour construire les algorithmes d'apprentissage automatique est le réseau de neurones. S'inspirant du neurone biologique, sa première forme, appelée le perceptron, fut modélisée en 1957 par ROSENBLATT [1958]. Il connut quelques développements les décennies suivantes, notamment avec les réseaux multi-couches entrainés par descente de gradient dans les années 80 par RUMELHART et collab. [1985] et LE CUN [1986]. Il ne prit réellement de l'ampleur que récemment grâce à l'augmentation des capacités de calcul et aux très bons résultats obtenus en classification d'image.

Couches de neurones

Le neurone est la brique fondamentale qui constitue ces outils, il peut être décrit comme une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ où n est la taille de l'entrée. Il est définit par une fonction non-linéaire f appelée fonction d'activation, ses poids $\mathbf{A} \in \mathbb{R}^n$ et son biais $b \in \mathbb{R}$, on peut alors lui associer une fonction $\phi_{f,\mathbf{A},b}$ qui donnera sa sortie en fonction de son entrée :

$$\begin{array}{rccc} \phi_{f,\mathbf{A},b}: & \mathbb{R}^n & \rightarrow & \mathbb{R} \\ & \mathbf{X} & \mapsto & f(\mathbf{A}.\mathbf{X}+b) \end{array} \tag{1.14}$$

Les neurones sont généralement organisés en couches, une couche de taille m est alors définie par sa fonction d'activation f, ses poids $\mathbf{A} \in \mathbb{R}^{m,n}$ et ses biais $\mathbf{B} \in \mathbb{R}^m$, on peut alors y associer, comme précédemment, une fonction :

$$\begin{array}{rccc} \Phi_{f,\mathbf{A},\mathbf{B}}: & \mathbb{R}^n & \rightarrow & \mathbb{R}^m \\ & \mathbf{X} & \mapsto & f(\mathbf{A}\mathbf{X}+\mathbf{B}) \end{array} \tag{1.15}$$

Dans un NN, on assemble ensuite les couches l'une après l'autre, la fonction d'activation non-linéaire est alors nécessaire puisque sans elle la fonction associée à une couche est affine et la composition de deux applications affines reste une application affine, enchaîner deux couches n'apporterait donc rien de plus au réseau. On peut parfois arranger les couches de manière un peu plus compliquée comme par exemple dans la cellule du LSTM sur lesquelles nous reviendrons plus tard. Pour les fonctions d'activation, la plus courante est la Rectified Linear Unit (ReLU) définie par $f(x) = max(0,x)$ ou une de ses variantes (ajout de biais, seuil, légère pente à gauche...). On peut aussi trouver des fonctions logistiques telles que sigmoïde ou softmax selon la forme que l'on compte imposer à la sortie de la couche.

1. https ://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Réseaux de neurones récurrents

Les réseaux de neurones récurrents sont une variante des réseaux classiques particulièrement adaptée au traitement de séquences (texte, séries temporelles...). L'idée est ici de se servir d'un réseau de neurones simple de façon récurrente. C'est-à-dire que pour chaque itération sur la séquence de données, ce réseau prendra en entrée la concaténation des valeurs de la séquence à cette itération et de sa propre sortie à l'itération précédente. Ainsi si on note $(X_t)_{t\in[|1,N|]}$ la séquence des données d'entrées, Ψ_θ le réseau de neurone à utiliser récursivement et $(h_t)_{t\in[|1,N|]}$ la sortie du RNN ainsi défini, alors en notant $[v_1, v_2]$ la concaténation des vecteurs v_1 et v_2 :

$$\forall t \in [|1,N|],\ h_t = \Psi_\theta([X_t, h_{t-1}])$$

L'initialisation h_0 de la cellule peut être aléatoire ou fixée suivant les besoins.

L'un des RNN les plus utilisés actuellement est le Long Short-Term Memory par HOCHREITER et SCHMIDHUBER [1997] (LSTM) dont une représentation peut être trouvée en figure 1.4. Il a la particularité d'avoir en plus de sa sortie h_t, une mémoire c_t. Il est constitué de quatre couches de neurones qui ont des rôles spécifiques, prenant en entrée à chaque fois la concaténation de la sortie précédente h_{t-1} et la nouvelle entrée X_t. Les trois premières ont pour tâche de modifier la mémoire. La toute première de sortie f_t (forget) a généralement pour fonction d'activation une fonction sigmoïde (entre 0 et 1) ses valeurs sont multipliées point par point à la mémoire, elle permet en fait d'oublier certaines informations. Les deux suivantes permettent d'écrire dans la mémoire, l'une de sortie $\tilde{c}_t$ représente l'information que l'on peut extraire, l'autre de sortie i_t (input) avec une fonction d'activation en sigmoïde permet de déterminer lesquelles de ces informations méritent d'être retenues. Le produit des sorties de ces deux couches est ensuite ajouté à la mémoire. La dernière couche de sortie o_t (output) toujours en sigmoïde permet de choisir ce qu'il faut exprimer de cette mémoire, la sortie h_t du réseau est donc le produit de la mémoire c_t et de o_t après être passé par une fonction de régularisation (ReLU, tanh, sigmoïde...). Les équations qui régissent le fonctionnement de la cellule LSTM sont les suivantes :

$$f_t = \sigma(W_f.[X_t, h_{t-1}] + B_f) \tag{1.16}$$
$$i_t = \sigma(W_i.[X_t, h_{t-1}] + B_i) \tag{1.17}$$
$$\tilde{c}_t = tanh(W_{\tilde{c}}.[X_t, h_{t-1}] + B_{\tilde{c}}) \tag{1.18}$$
$$o_t = \sigma(W_o.[X_t, h_{t-1}] + B_o) \tag{1.19}$$
$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{1.20}$$
$$h_t = tanh(o_t \times c_t) \tag{1.21}$$

où . représente le produit matriciel, $\times$ la multiplication point par point et $[]$ l'opération de concaténation. W et B sont respectivement les poids et biais des couches de neurones qui composent la cellule.

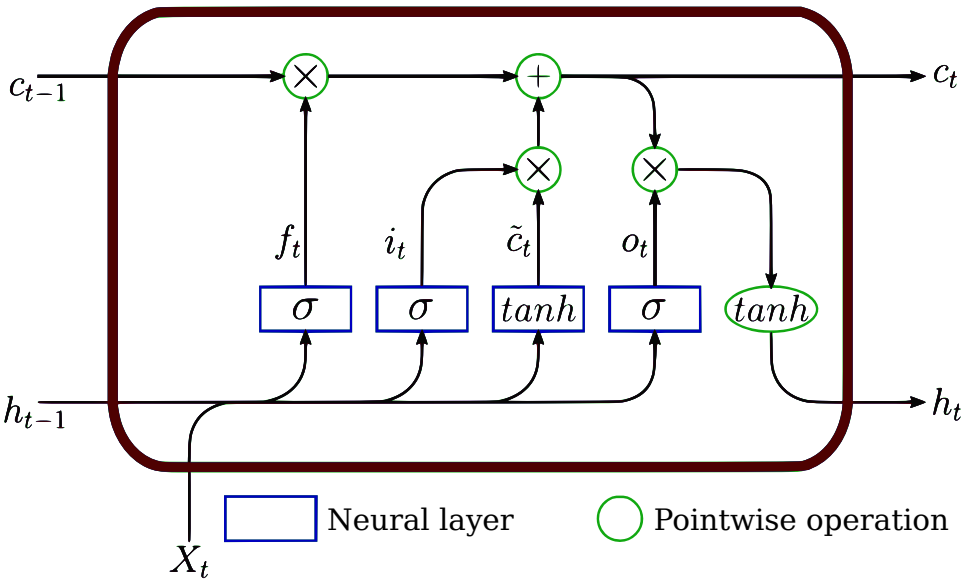


FIGURE 1.4 – Schéma de fonctionnement du LSTM. X_t représente les données reçues en entrée par la cellule à l'instant t, c_t et h_t représentent respectivement la mémoire et la sortie exprimée par la cellule à l'issue de cette nouvelle entrée. Les couches de neurones en σ sont appelées des portes, leur sorties notées f_t (*forget*), i_t (*input*) et o_t (*output*) ne contiennent que des éléments entre 0 et 1 et ont des rôles spécifiques visant à effacer ou exprimer certaines informations.

Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN) étaient à l'origine destinés à des tâches de classification d'image et s'inspiraient de l'organisation des neurones biologiques dans le cortex visuel. L'idée est d'appliquer une convolution discrète en 2D entre une image et un noyau entraînable. De la même manière, une convolution discrète en 1D peut être appliquée sur les séries temporelles. Un filtre est alors défini comme pour les couches de neurones par son noyau, son biais et sa fonction d'activation. Plusieurs filtres peuvent être superposés pour extraire différentes informations à partir des mêmes entrées multicanaux. La transformation effectuée par un filtre de convolution peut s'écrire :

$$\mathrm{Y}_i = f\Big(\sum_{k=0}^{n_f-1} (\mathrm{W}_{i,k} * \mathrm{X}_k) + \mathrm{B}_i\Big) \tag{1.22}$$

où $*$ représente l'opération de convolution, Y_i l'expression du $i^{\text{ème}}$ filtre du CNN, n_f le nombre de canaux du signal d'entrée X, X_k son $k^{\text{ème}}$ canal, W et B le noyau et le biais de la couche de convolutions, et f sa fonction d'activation.

1.3.3 Hybridation avec des filtres de Kalman

L'approche que nous avons choisie est de ne pas se priver de tout ce qui a déjà été fait sur les filtres de Kalman en pistage, et qui fonctionne très bien lorsque le filtre est bien réglé. Nous avons plutôt choisi de travailler sur des filtres hybrides qui intègrent des réseaux de neurones à un filtre de Kalman. Dans cette partie, nous développons les quelques approches qui ont été faites dans ce sens.

Deep Kalman

En 2015, KRISHNAN et collab. [2015] ont pour la première fois à notre connaissance intégré des réseaux de neurones dans un filtre de Kalman. Leur idée était de remplacer l'étape de prédiction de l'espérance et de la covariance du vecteur d'état par l'utilisation de réseaux entraînables. Ils ont donc créé un filtre de Kalman qui apprenait de lui-même son modèle d'évolution.

Backprop Kalman Filter

En 2016, HAARNOJA et collab. [2016] introduisent leur Backprop Kalman Filter. Le principe est de combiner un algorithme d'apprentissage profond dont l'objectif est d'évaluer une mesure sous la forme d'une espérance et d'une matrice de covariance à un filtre de Kalman. Un schéma présent en figure 1.5 montre le fonctionnement de ce filtre. Le réseau proposé enchaîne deux couches de convolutions puis deux couches simples, toutes les fonctions d'activation sont des ReLU, les convolutions sont suivies d'une étape de sous-échantillonnage en max pooling, et une étape de normalisation précède les fonctions d'activation des couches de convolution. À la suite de ces quatre couches, l'information est dupliquée et envoyée à deux nouvelles couches simples sans fonction d'activation. La première donnera l'espérance de la mesure et l'autre, après un réarrangement, une matrice triangulaire qui une fois les éléments diagonaux passés à l'exponentielle est multipliée à sa transposée pour obtenir une matrice symétrique définie positive qui servira de matrice de covariance pour le bruit de mesure. Le réseau est ensuite entraîné pour chercher à minimiser la moyenne des carrés des erreurs à la sortie du filtre de Kalman. Ce prétraitement de l'information permet de faire fonctionner un filtre de Kalman sur des informations de hautes dimensions, comme par exemple un flux vidéo.

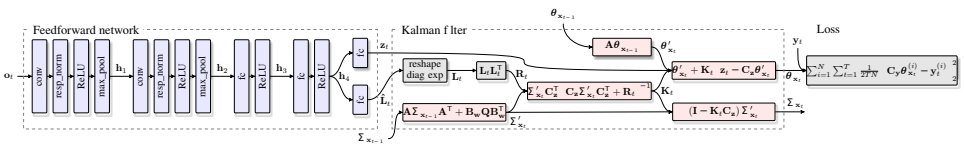


FIGURE 1.5 – Schéma de fonctionnement du Backprop Kalman Filter, issu de HAARNOJA et collab. [2016]

LSTM Kalman Filter

En 2017, COSKUN et collab. [2017] publient un filtre qui combinent les deux approches précédentes, le LSTM Kalman Filter. Le schéma de son fonctionnement se trouve en figure 1.6. L'idée est de partir d'un algorithme d'apprentissage profond qui effectue une tâche image par image (il peut s'agir d'une tâche de pistage, mais aussi de pose, de segmentation ou autre) et de transmettre cette première estimation à un filtre de Kalman pour exploiter la continuité temporelle de la vidéo. Ce premier réseau (*black-box estimator* sur le schéma) renvoie une mesure simple qui fera office d'espérance, mais pas de matrice de covariance de bruit. Un premier LSTM prend en entrée cette estimation pour sortir une matrice qui servira de bruit de mesure. L'opérateur d'évolution est lui aussi remplacé par un LSTM dont la matrice jacobienne sera utilisée pour l'évolution de la matrice de covariance de l'état, comme pour l'EKF ou le Deep Kalman. Le bruit de modèle est lui aussi déterminé par un LSTM qui se base sur l'état prédit. Les matrices de bruits de mesure et de modèle sont ici imposées diagonales avec passage à l'exponentielle pour qu'elles soient symétriques définies positives.

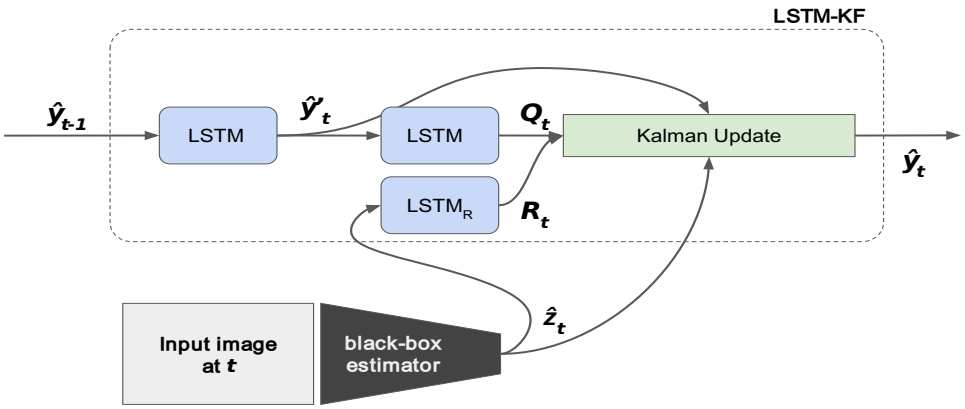


FIGURE 1.6 – Schéma de fonctionnement du LSTM Kalman Filter, issu de COSKUN et collab. [2017]

Interaction-aware Kalman Neural Networks

En 2019, JU et collab. [2019] ont proposé leur propre filtre. Il reprend la structure du LSTM Kalman Filter, mais l'étape de prédiction prend aussi en compte l'état des autres cibles pour anticiper leurs interactions.

1.4 Organisation et contributions de la thèse

1.4.1 Contributions

Lorsqu'on est en présence d'une cible manœuvrante suivant un modèle cinématique non-linéaire avec des paramètres inconnus, on peut aujourd'hui envisager deux grandes approches. La première consiste à modéliser au mieux les équations cinématiques dans le filtre, mais les paramètres des modèles complexes sont difficiles à régler. La seconde consiste à utiliser un filtre de type IMM avec de nombreux modèles cinématiques simples (mouvement rectiligne uniforme, accélération constante, virages exacts avec différentes courbures) et des bruits de modèle réglés de façon optimale sur les données simulées.

Pour des cinématiques complexes (non linéaires) avec des paramètres non maîtrisés, on peut penser que les techniques qui empruntent à l'IA offrent des pistes prometteuses. Si on a des simulations suffisamment représentatives de tous les mouvements auxquels le radar sera confronté, on peut penser que l'IA serait plus à même de trouver quels modèles cinématiques sont prépondérants dans le filtre IMM et doivent être utilisés, comment les régler et comment régler les probabilités de transition, ce qui devient vite une tâche difficile pour l'ingénieur quand le nombre de modèles augmente. C'est la piste que nous avons exposée au cours de cette thèse.

Une **première contribution** principale de cette thèse consiste à utiliser des méthodes de type réseaux de neurones profonds pour faire de la classification en temps réel de cibles à partir des mesures de leur position, et des paramètres cinématiques qu'on peut inférer à partir de ces dernières (vitesse, courbure, torsion des trajectoires). L'apprentissage est effectuée à partir de vraies données ADS-B transmis par les appareils, que nous avons récupérées, puis les résultats sont évalués sur une partie de ces données réservées pour la validation et montrent la granularité de classification des aéronefs que l'on peut espérer à partir uniquement de leur trajectoire.

Notre **deuxième contribution** principale consiste à étudier le réglage adaptatif d'un filtre de Kalman basé sur un modèle rectiligne uniforme au travers d'un réseau de neurones ayant appris sur un grand nombre de simulations les meilleurs réglages.

Finalement, notre **troisième contribution** principale consiste à considérer un filtre de type IMM basé sur un grand nombre de modèle, et à utiliser une base importante de trajectoires simulées pour apprendre quels modèles doivent être utilisés, le réglage de leurs bruits de modèles, ainsi que les matrices de transition.

Dans ces deux derniers cas, il convient de savoir simuler un grand nombre de trajectoires pour entraîner les réseaux de neurones. En effet, les données radar sont difficiles à collecter, surtout lorsqu'il y a des enjeux de confidentialité, et on ne peut pas espérer en disposer d'un nombre suffisant pour entraîner des réseaux profonds. Les techniques de production de trajectoires simulées consti-

tuent aussi une **contribution**, plus mineure, de cette thèse.

1.4.2 Organisation du document

Cette thèse est séparée en quatre chapitres.

Le Chapitre 1 introduit le sujet ainsi que l'état de l'art spécifique aux problèmes qu'il soulève.

Le Chapitre 2 est consacré à la classification d'aéronefs. Différents algorithmes de réseaux de neurones profonds sont utilisés sur des trajectoires réelles d'appareils obtenues par lissage des positions transmises par le protocole ADS-B.

Le Chapitre 3 est consacré à l'étude des filtres adaptatifs. Une adaptation par réseau de neurones baptisée NNAKF est proposée et comparée à d'autres filtres de la littérature. Les entraînements et analyses sont effectués sur trajectoires simulées.

Le Chapitre 4 est consacré à l'étude des filtres à modèles multiples. L'IMM est implémenté et une version intégrant une cellule LSTM est proposée. Les entraînements et analyses sont effectuées sur les mêmes trajectoires simulées qu'au Chapitre 3.

Le Chapitre 5 reprend les conclusions générales de la thèse.

Les annexes sont au nombre de trois et incluent :

— la génération des trajectoires simulées utilisées dans les Chapitres 3 et 4 pour l'apprentissage.

— la formulation des fonctions de coût utilisées pour l'apprentissage des algorithmes de pistage.

— les définitions des modèles cinématiques utilisés par les algorithmes de pistage.

1.4.3 Publications

Conférences

— Pilté, M., Jouaber, S., Bonnabel, S., Barbaresco, F., Fragu, M., & Honoré, N. (2019, September). Invariant extended kalman filter applied to tracking for air traffic control. In *2019 International Radar Conference (RADAR)* (pp. 1-6). IEEE.

— Jouaber, S., Bonnabel, S., Velasco-Forero, S., & Pilté, M. (2021, June). NNAKF : A Neural Network Adapted Kalman Filter for Target Tracking. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4075-4079). IEEE.

— Jouaber, S., Bonnabel, S., Pilté, M., & Velasco-Forero, S. (2021, December). Neural Network Adapted Kalman Filter. In *IAMD 2021 International Conference on Integrated Air and Missile Defence*.

Journaux

— Jouaber, S., Bonnabel, S., Velasco-Forero, S., Pilté, M., & Angulo, J. (soumission en cours). Real-time classification of aircrafts manoeuvers. In *Journal of Signal Processing Systems*.

Brevets

— Chopin, P., Barbaresco, F., & Jouaber, S. (2020, October). Device for identifying a type of aircraft, associated identification method and computer program. In *EP Patent EP3722830A1*.

— Jouaber, S., Pilté, M., Bonnabel, S., & Velasco-Forero, S. (dépôt en cours). Dispositif de suivi d'une évolution d'au moins un aéronef dans un espace aérien, système de surveillance et procédé de suivi associés.

1.5 Références

BLANCHARD, Y. 2004, *Le radar, 1904-2004 : Histoire d'un siècle d'innovations techniques et opérationnelles*, Ellipses. 2

BLOM, H. A. et Y. BAR-SHALOM. 1988, «The interacting multiple model algorithm for systems with markovian switching coefficients», *IEEE transactions on Automatic Control*, vol. 33, nº 8, p. 780–783. 9, 10

COSKUN, H., F. ACHILLES, R. DIPIETRO, N. NAVAB et F. TOMBARI. 2017, «Long short-term memory kalman filters : Recurrent neural estimators for pose regularization», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 5524–5532. 16

CURRY, G. R. 2005, «Radar system performance modeling», . 4

DARRICAU, J. 2002, *Physique et théorie du radar : principes et éléments de base*, Éd. Deniaud. 2

HAARNOJA, T., A. AJAY, S. LEVINE et P. ABBEEL. 2016, «Backprop kf : Learning discriminative deterministic state estimators», dans *Advances in Neural Information Processing Systems*, p. 4376–4384. 15

HOCHREITER, S. et J. SCHMIDHUBER. 1997, «Long short-term memory», *Neural computation*, vol. 9, n° 8, p. 1735–1780. 13

JU, C., Z. WANG, C. LONG, X. ZHANG, G. CONG et D. E. CHANG. 2019, «Interaction-aware kalman neural networks for trajectory prediction», *arXiv preprint arXiv :1902.10928*. 16

KINGMA, D. P. et J. BA. 2014, «Adam : A method for stochastic optimization», *arXiv preprint arXiv :1412.6980*. 12

KRISHNAN, R. G., U. SHALIT et D. SONTAG. 2015, «Deep kalman filters», . 15

LE CUN, Y. 1986, «Learning process in an asymmetric threshold network», dans *Disordered systems and biological organization*, Springer, p. 233–240. 12

ROSENBLATT, F. 1958, «The perceptron : a probabilistic model for information storage and organization in the brain.», *Psychological review*, vol. 65, n° 6, p. 386. 12

RUMELHART, D. E., G. E. HINTON et R. J. WILLIAMS. 1985, «Learning internal representations by error propagation», cahier de recherche, California Univ San Diego La Jolla Inst for Cognitive Science. 12

SKOLNIK, M. I. 1970, «Radar handbook», . 2

ZEILER, M. D. 2012, «Adadelta : an adaptive learning rate method», *arXiv preprint arXiv :1212.5701*. 12

Chapitre 2

Classification d'aéronefs

Sommaire

2.1 Résumé

Que ce soit pour des applications de défense aérienne ou de contrôle du trafic, il est souhaitable d'estimer en temps réel le type d'aéronef auquel on a affaire. Ce processus peut s'avérer utile lorsque la cible refuse de coopérer ou pour confronter les informations transmises avec la trajectoire observée. Dans ce chapitre, nous proposons une approche basée uniquement sur des mesures de position (radar) pour des raisons d'adaptabilité aux contextes réels. De plus, comme attributs distinctifs pour la classification automatique en temps réel de trajectoires d'aéronefs, nous proposons de nous concentrer sur des grandeurs cinématiques invariantes par rotation et translation dans le plan horizontal, comme la vitesse, la courbure, ou la torsion. Ces valeurs sont utilisées par des réseaux neuronaux convolutifs à l'état de l'art pour des tâches de classification. Ceux-ci ont la particularité de pouvoir traiter sans problème des trajectoires de longueurs variables et peuvent fonctionner en temps réel. Les classifieurs étudiés sont entraînés avec des données réelles que nous avons collectées grâce aux informations transmises publiquement par les aéronefs. Ceci nous permet de comparer les différents algorithmes appris, et de discuter des limites de précision qu'ils peuvent atteindre.

2.2 Introduction

Afin d'améliorer la sécurité du trafic aérien, il est utile de trouver un moyen de déterminer en temps réel le type d'appareils en vol. En particulier, l'augmentation du nombre de drones intégrés à l'espace aérien civil, que ce soit pour des raisons commerciales, ludiques, ou éventuellement malveillantes, nous poussent à envisager le développement de tels algorithmes. De l'identification du type d'aéronef découlent des informations sur ses capacités de manœuvre, et potentiellement à terme l'évaluation de la menace qu'il peut représenter.

Le problème de la classification des aéronefs basée sur des mesures radar a été traité par le passé grâce aux profils de distance des échos dans YAN et collab. [1997]; ZYWECK et BOGNER [1996], aux spectres micro-Doppler dans LEI et LU [2005]; MOLCHANOV et collab. [2014], ou à l'imagerie des Radar à Synthèse d'Ouverture (RSO/SAR) dans CHEN et collab. [2016]; NASRABADI [2019]; PIERUCCI et BOCCHI [2007]; THIAGARAJAN et collab. [2010]. Cependant, ces types de mesures peuvent ne pas toujours être disponibles. Dans ce papier, nous nous concentrons sur une classification basée uniquement sur les mesures de position accessibles au travers d'un simple radar.

À notre connaissance, le problème de la classification en temps réel de cible volante a été pour la première fois par GHADAKI et DIZAJI [2006] et leur invention a été brevetée dans DIZAJI et GHADAKI [2006]. Leur objectif principal était de distinguer les aéronefs et des cibles biologiques comme les oiseaux, la dif-

ficulté étant qu'un hélicoptère peut voler aussi lentement qu'un oiseau. Leur deuxième objectif était de détecter les appareils pour lesquels manquaient les données de radars secondaires (SSR). Leur approche s'est par la suite spécialisée dans la détection de drones dans l'espace aérien civil dans MOHAJERIN et collab. [2014]. La méthode proposée consistait à extraire des grandeurs depuis les mesures comme la vitesse moyenne et la distance totale parcourue, et à classifier les cibles à l'aide d'un jeu de données d'entraînement et d'un classifieur standard, comme un perceptron multicouche (MLP) ou une machine à vecteurs de support (SVM).

D'un point de vue général, notre approche diffère de ces précédents travaux sur au moins deux aspects importants. D'abord, nous cherchons à répondre à un problème différent qui consiste à estimer un certain nombre de caractéristiques propres aux aéronefs sans nous concentrer sur les petites cibles comme les drones et les oiseaux. Ensuite, nous exploitons la littérature récente concernant la classification de séries temporelles utilisant des méthodes avancées d'apprentissage profond comme les réseaux de neurones convolutifs (FCN) ou les réseaux résiduels (ResNet), voir FAWAZ et collab. [2019]; WANG et collab. [2017]. D'un point de vue plus technique, nous prônons l'usage de grandeurs cinématiques différentes pour décrire les trajectoires. En particulier, nous nous appuyons sur la même idée clé que NABAA et BISHOP [2000] que la torsion est une caractéristique discriminante pour les cibles rapides comme les avions de chasse.

Nos principales contributions peuvent être résumées ainsi :

— Afin de traiter le problème de la classification en temps réel d'aéronefs à partir de mesures de position, nous avons collecté des transmissions ADS-B (Automatic Dependent Surveillance-Broadcast) pour créer un jeu de données.

— Pour alimenter les classifieurs, nous avons choisi d'utiliser des grandeurs cinématiques invariantes par rotation et translation dans le plan horizontal, la vitesse, la courbure, la torsion, l'altitude et ses dérivées temporelles. Toutes ces valeurs peuvent être extraites directement après une étape de lissage et d'interpolation basée sur une régression quadratique locale des trajectoires, rendant l'algorithme robuste aux variations de pas de temps et compatible avec un traitement en temps réel.

— Trois différents classifieurs issus de la littérature récente sur les réseaux de neurones convolutifs appliqués aux séries temporelles sont proposés pour classifier les aéronefs à partir des grandeurs extraites : ils expriment tous les probabilités pour chaque cible d'appartenir à chacune des classes. Leur fonctionnement est compatible avec une utilisation en temps réel et ne dépend pas de la durée des trajectoires.

— L'approche consistant à collecter des données ADS-B et à retenir les caractéristiques qui y sont associées pour définir les classes d'aéronefs étant nouvelle à notre connaissance, nous discutons de la limite de précision

atteignable avec ce genre de méthodes en fonction de la granularité de la classification étant donné que différents types d'aéronefs peuvent avoir des trajectoires presque identiques.

La méthodologie proposée ici a fait l'objet d'un brevet déposé par Thales CHOPIN et collab. [2020-10-14] étant donné qu'elle a des applications dans la régulation du trafic aérien, et est en cours de soumission au *Journal of Signal Processing Systems*.

Le reste du chapitre est organisé de la façon suivante. La section 2 est dédiée à la création des jeux de données. Dans la section 3 nous discutons des grandeurs cinématiques utilisées pour la classification des trajectoires, et l'algorithme servant à les extraire est présenté. La section 4 résume la littérature en lien avec l'utilisation de réseaux de neurones pour des problèmes de classification de séries temporelles. De là, nous proposons et détaillons les classifieurs à réseaux de neurones convolutifs dans la section 5. Enfin, la section 6 est dédiée aux résultats expérimentaux et à la discussion.

2.3 Problème et littérature spécifique

2.3.1 Données collectées

Grâce au protocole ADS-B, avions, hydravions, hélicoptères, autogires, aéronefs à aile basculante, ainsi que certains véhicules au sol diffusent leur identité ainsi que leur position donnée par les satellites de navigation. Cette identité est ensuite associée à une piste radar existante. Ces informations sont diffusées publiquement et peuvent être réceptionnées à l'aide d'une antenne appropriée. Elles peuvent par la suite servir de jeu de données labellisé pour être utilisé par des algorithmes d'apprentissage supervisé, comme nous le faisons.

Pour ce faire, nous avons utilisé la plateforme *ADS-B Exchange*[1] pour récupérer les positions des aéronefs dans une zone de $1500km$ de rayon en Europe pendant plusieurs heures. En plus de la position GPS et de l'altitude, chaque diffusion est associée à un aéronef par son identifiant ICAO (*International Civil Aviation Organization*), sa trajectoire au cours du temps peut donc être extraite. Les données de la diffusion ADS-B incluent aussi certaines spécification propres à l'aéronef, comme son modèle, son espèce (avion ou hélicoptère), sa WTC (*Wake Turbulence Category*) qui reflète la taille de l'appareil, son type de moteur (piston, turbopropulseur, réaction) et le nombre de ces derniers. Ces quatre dernières caractéristiques sont utilisées pour labéliser les aéronefs avec un label à quatre chiffres comme décrit dans le Tableau 2.1. Par exemple, grâce au tableau, on peut voir que le label "1231" fait référence à un avion de taille moyenne équipé d'un unique moteur à réaction, ce qui correspond généralement aux avions de chasse (étonnamment il y avait suffisamment de ceux-ci

1. www.adsbexchange.com

TABLEAU 2.1 – Description des labels du jeu de données

Position du chiffre	Spécification correspondante	Description
#1	Espèce	1 : avion 4 : hélicoptère
#2	WTC	1 : léger 2 : moyen 3 : lourd
#3	Type de moteur	1 : piston 2 : turbopropulseur 3 : réaction
#4	Nombre de moteurs	entier

diffusant publiquement leurs données ADS-B pour avoir une classe correspondante dans notre jeu de données). Le label sera par la suite exprimé par un encodage *one-hot* dans le jeu de données comme étant la sortie attendue des classifieurs. Après acquisition, 19 labels différents étaient représentés par plus de 50 trajectoires, les autres étant représentés par moins de 20 trajectoires. Ainsi, nous avons sélectionné 50 trajectoires pour chacun des 19 labels pour construire notre jeu de données. Ce faisant, nous avons choisi de rejeter aléatoirement certaines trajectoire pour balancer les classes, afin d'éviter en particulier la surreprésentation du label "1232" qui regroupe la plupart des avions de ligne.

Un exemple de données brutes collectées est représenté Figure 2.1. Les données de positions, mais aussi beaucoup l'horodatage, sont fortement bruitées car elles proviennent d'un grand nombre de sources avec des défauts différents. Pour cette raison, et aussi parce que nous voulons imposer un pas de temps régulier dans le jeu données final pour permettre aux réseaux neuronaux convolutifs de fonctionner dans de bonnes conditions, ces données ont été pré-traitées. Dans un premier temps les coordonnées GPS mesurées ont été converties en coordonnées cartésiennes en mètres dans un repère recentré sur chaque trajectoire. Ensuite la position sur chacun des trois axes à n'importe quel temps t, ainsi que les différentes grandeurs cinématiques issues de leurs dérivées temporelles, sont calculées à l'aide d'un algorithme de lissage que nous allons maintenant présenter.

2.3.2 Lissage de trajectoires et extraction des grandeurs cinématiques

Notre but est de proposer un algorithme capable d'inférer la catégorie de l'aéronef uniquement à partir de ses mesures de position, comme obtenues habituellement des radars. Pour cela, nous respecterons les contraintes suivantes :

1. l'algorithme doit être invariant aux translations et rotations horizontales
2. l'algorithme doit pouvoir gérer des trajectoires de durées différentes

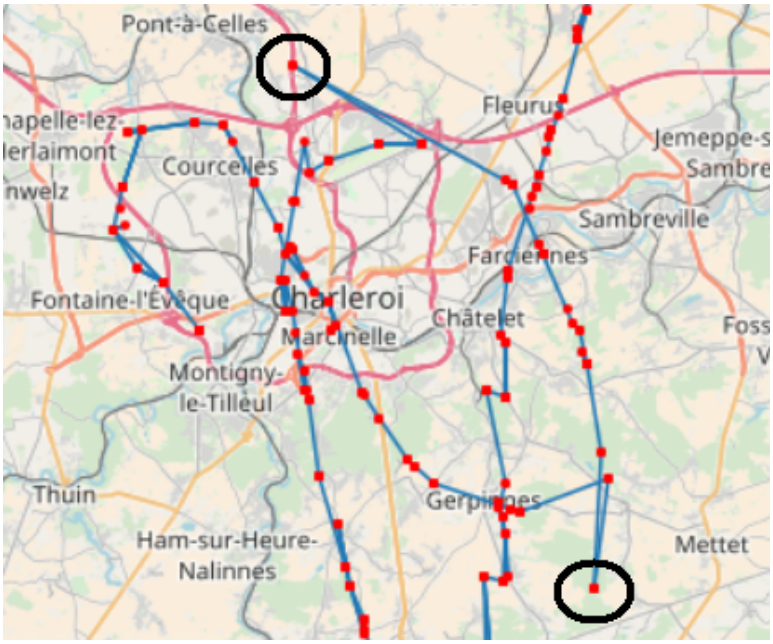


FIGURE 2.1 – Exemple de trajectoire brute, un avion de chasse belge F-16 manœuvre autour de Charleroi. Les positions reçues sont représentées par des carrés rouges, les positions consécutives sont reliées par une ligne bleue. Certains points semblent très écartés de la trajectoire réelle, surtout les deux cerclés de noir, révélant un bruit important dans les données.

3. l'algorithme associe une probabilité à chaque catégorie dans laquelle l'aéronef observé peut être classé

La première contrainte concerne la physique du problème, il n'y a pas de raison pour qu'une trajectoire dirigée vers le nord ou l'est soit privilégiée pour identifier la classe de l'aéronef, l'algorithme se doit donc de ne tenir compte que du mouvement. La seconde a déjà été mentionnée en introduction et vise à contraindre l'algorithme à pouvoir fonctionner en temps réel. Enfin la dernière a pour but l'interprétabilité des résultats, étant donné que certaines trajectoires peuvent être parfaitement indiscernables bien que relevant d'appareils de catégories différentes.

Afin de respecter ces trois contraintes, ainsi que pour renforcer la robustesse aux bruits de mesures et rendre le jeu de données à pas de temps constant grâce à un algorithme de lissage, nous approximons localement la trajectoire par une fonction quadratique. Dans ce but, nous effectuons la régression des trois coefficients qui définissent l'approximation quadratique à l'aide d'un noyau gaussien de rayon τ comme suit :

$$x(t) = \left[\underset{a,b,c}{\arg\min} \sum_i e^{\frac{-(t-t_i)^2}{2\tau^2}} \left(\begin{bmatrix} a \\ b \\ c \end{bmatrix} . \begin{bmatrix} t_i^2 \\ t_i \\ 1 \end{bmatrix} - x_i \right)^2 \right] . \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix}$$

Les paramètres a, b, c sont facilement calculables par les formules standards d'approximation par moindre carrés, et le paramètre τ a été fixé à $5s$ pour notre jeu de données. Cette méthode permet non seulement d'approximer la position de l'aéronef à n'importe quel temps t, mais aussi la vitesse, la courbure et la torsion de sa trajectoire par différentiation. Pour respecter la première contrainte évoquée précédemment, seules les grandeurs invariantes par translation et rotation dans le plan horizontal, soit la vitesse, la courbure, la torsion, l'altitude et ses trois premières dérivées. Pour chaque trajectoire, dix segments de dix minutes de long ont été sélectionnés et un échantillonnage à $0.1s$ de pas de temps à été fait pour calculer ces grandeurs.

Ce processus aboutit au final à un jeu de données de dimension $(9500 \times 6001 \times 7)$ à fournir à l'entrée du classifieur et de dimension (9500×19) pour l'expression attendue de la classification. Pour la phase d'apprentissage, le jeu de données a été coupé en deux, 70% des trajectoires ont été utilisés pour le calcul du gradient et l'optimisation du classifieur, et 30% ont été utilisés pour surveiller l'évolution de la précision du classifieur. Pour garder ces deux parties du jeu de données indépendantes, les trajectoires correspondant au même aéronef ont été conservées ensemble.

2.3.3 Les réseaux de neurones utilisés pour les problèmes de classification

Dans cette section, nous passerons rapidement en revue les réseaux de neurones utilisés pour des tâches de classification de séries temporelles dans la littérature en portant une attention particulière à ceux qui pourront être pertinents pour notre problème.

Concernant les algorithmes de classification, les solutions à l'état de l'art sont généralement des algorithmes de plus proches voisins basés sur l'Adaptation Temporelle Dynamique ou une de ses variantes JEONG et collab. [2011]; RATANAMAHATANA et KEOGH [2004]; XI et collab. [2006] comme alternative à la distance euclidienne. Cependant le temps d'exécution de ce genre d'algorithmes est souvent trop long pour une classification en temps réel. C'est pourquoi nous nous intéresserons ici plutôt à des solutions basées sur les réseaux de neurones profonds. L'une des principales restrictions que les algorithmes employés doivent respecter est qu'il doivent pouvoir fonctionner avec des trajectoires de longueurs variables. En termes de réseaux de neurones, différentes architectures peuvent être utilisées. Les approches les plus simples impliquent des couches de neurones basiques comme proposées par WANG et collab. [2017], mais ces solutions requièrent des trajectoires de longueur fixe en entrée ce qui n'est pas compatible avec notre problème. Une autre solution est d'utiliser des

réseaux de neurones récurrents, comme le Time Warping Invariant Echo State Network (TWIESN) proposé par TANISARO et HEIDEMANN [2016], mais la plupart des solutions reposent sur les réseaux de neurones convolutifs LE GUENNEC et collab. [2016]; ZHAO et collab. [2017]. Ces derniers utilisent des noyaux de petites tailles qui agissent comme des filtres pour extraire des information locales directement depuis les séries temporelles. Il est aussi possible d'augmenter manuellement les données en ajoutant un pré-traitement, comme pour le Multi-scale Convolutional Neural Network (MCNN) proposé par CUI et collab. [2016] et sur lequel nous reviendrons. Après les couches de convolution, une étape de *pooling* globale, généralement un maximum ou une moyenne, permet d'agréger l'information dans la dimension temporelle pour obtenir un vecteur de taille fixe qui peut être employé par un réseau de neurones classique qui effectuera la classification WANG et collab. [2017]; ZHENG et collab. [2014] : ceci permet au classifieur de fonctionner sans contrainte sur la taille des trajectoires. Une autre alternative que nous n'explorerons pas ici est l'utilisation de mécanismes d'attention comme dans SERRÀ et collab. [2018].

2.4 classifieurs envisagés

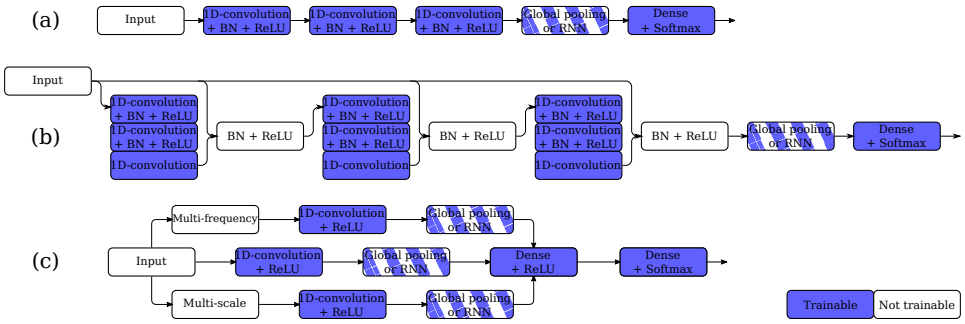


FIGURE 2.2 – Schémas des différents classifieurs, FCN (a), ResNet (b) et MCNN (c).

Dans cette section, nous décrivons les trois architectures de réseaux de neurones entraînés pour résoudre notre problème de classification. Ces architectures sont schématisés en figure 2.2. En 2019, Fawaz *et al.* FAWAZ et collab. [2019] ont comparé différents classifieurs sur des problèmes de classification de séries temporelles. Parmi ceux-ci nous avons choisi de garder le Fully Convolutional Neural Network (FCN) et le Residual Network (ResNet) proposés par Wang *et al.* WANG et collab. [2017] puisqu'ils étaient en haut du classement pour la plupart des problèmes envisagés. Nous avons aussi testé le Multi-scale Convolutional Neural Network (MCNN) proposé par Cui *et al.* CUI et collab. [2016].

2.4.1 Fully-Convolutional Neural Network

Le FCN, représenté dans la figure 2.2 (a) possède une architecture relativement simple. Il est constitué de trois couches de convolution avec des Batch Normalization (BN) et des Rectified Linear Unit (ReLU) pour fonctions d'activation. La BN IOFFE et SZEGEDY [2015] est généralement utilisée pour accélérer le processus d'apprentissage et parfois éviter le sur-apprentissage. Ensuite, une couche de *pooling* ou un RNN est utilisé pour agréger la série dans le temps en un vecteur de taille fixe, qui est lui fourni en entrée à une couche de neurones muni d'une fonction d'activation *softmax*. Un *Global Max Pooling* (GlobalMaxPool) ou une cellule LSTM peuvent être utilisées pour l'agrégation, la différence étant que la cellule LSTM est capable de retenir et d'oublier de l'information au besoin, tandis qu'une forte valeur à la sortie de la dernière couche de convolution à un instant donné pourra affecter la sortie d'un GlobalMaxPool pour tout le reste de la trajectoire. En utilisant une fonction d'activation *softmax* à la fin du classifieur, ce dernier exprimera à l'issue de chaque nouvelle donnée d'entrée un vecteur pouvant être interprété comme la probabilité pour l'aéronef d'appartenir à chacune des classes possibles.

2.4.2 Residual Network

Le ResNet, représenté dans la figure 2.2 (b), est constitué de trois blocs chacun constitué de trois couches de convolution, chacune utilisant BN et ReLU comme le FCN. À la fin de chaque bloc, entre la dernière convolution et la BN qui suit, l'entrée brute du filtre est concaténée, c'est ce que l'on appelle une *skip connection*. Cette architecture était à l'origine destinée à des problèmes de reconnaissance d'image par He *et al.* HE et collab. [2016], les *skip connections* permettent d'atténuer la dégradation de l'information au sein des réseaux de neurones profonds. À l'issue des trois blocs, les informations sont agrégées dans le temps et une couche de neurones munie d'une fonction d'activation *softmax* permet d'exprimer le résultat de la classification, exactement comme pour le FCN.

2.4.3 Multi-scale Convolutional Neural Network

Le MCNN, représenté en figure 2.2 (c), inclut une étape de pré-traitement de l'entrée brute avant de fournir une version augmentée de l'information aux CNNs. Dans le bloc *Multi-frequency*, des moyennes glissantes de différentes longueurs sont utilisées comme filtres passe-bas avec différentes fréquences de coupure. Dans le bloc *Multi-scale*, des *poolings* de différentes longueurs sont utilisées pour que les convolutions suivantes travaillent sur différentes échelles de temps. Les sorties de ces deux blocs ainsi que l'entrée brute sont ensuite fournies à des CNNs dont les sorties sont ensuite agrégées dans le temps, et finalement fournies à deux couches de neurones munies d'une fonction d'activation ReLU pour la première et *softmax* pour la seconde.

TABLEAU 2.2 – Résumé des différents classifieurs avec leur nombre de paramètres entraînables, leur précision sur le jeu de données de test après entraînement ainsi que leur précision sur deux problèmes restreints : classifier l'espèce de l'aéronef (avion ou hélicoptère), et classifier la WTC des avions.

Architecture	Agrégation	Hyper-paramètres (k_1, k_2, k_3)	Paramètres	Précision sur 19 classes (%)	Précision sur l'espèce (%)	Précision sur la WTC des avions (%)
FCN	Max Pooling	5, 20, 5	53 075	34.81	87.23	68.71
		1, 9, 20	60 371	33.93	89.05	69.24
		20, 9, 1	25 715	35.61	88.07	67.51
FCN	LSTM	5, 20, 5	78 515	32.84	87.16	70.49
		1, 9, 20	85 811	30.91	87.82	67.42
		20, 9, 1	51 155	32.95	88.91	68.04
ResNet	Max Pooling	1, 13, 1	114 995	33.72	**90.00**	67.73
		1, 5, 9	114 995	32.74	88.70	70.40
		9, 5, 1	98 739	35.23	88.28	67.64
ResNet	LSTM	1, 13, 1	85 891	35.65	88.18	70.44
		1, 5, 9	85 891	32.39	87.61	68.31
		9, 5, 1	64 259	34.00	87.61	69.64
MCNN	Max Pooling	$n_c, n_d, n_f = 15, 50, 6$	26 879	31.16	87.72	65.29
		$n_c, n_d, n_f = 30, 50, 4$	46 739	31.82	87.79	66.58
		$n_c, n_d, n_f = 15, 100, 6$	37 629	32.81	88.21	66.93
MCNN	LSTM	$n_c, n_d, n_f = 15, 50, 4$	25 679	29.79	87.26	66.84
		$n_c, n_d, n_f = 30, 50, 6$	59 939	28.14	86.00	64.80
		$n_c, n_d, n_f = 30, 50, 4$	53 939	29.68	87.37	66.53
MCNN+BN	Max Pooling	$n_c, n_d, n_f = 15, 50, 6$	26 879	**39.86**	88.39	73.82
		$n_c, n_d, n_f = 30, 50, 4$	46 739	35.30	88.11	71.96
		$n_c, n_d, n_f = 15, 100, 6$	37 629	39.02	88.67	**74.13**
MCNN+BN	LSTM	$n_c, n_d, n_f = 15, 50, 4$	25 679	32.53	88.95	72.04
		$n_c, n_d, n_f = 30, 50, 6$	59 939	34.98	89.75	72.89
		$n_c, n_d, n_f = 30, 50, 4$	53 939	35.33	88.84	70.27

Pour chacun de ces algorithmes, le processus d'entraînement consiste à minimiser une fonction de coût donnée en ajustant les paramètres entraînables à l'aide d'un algorithme d'optimisation par descente de gradient fonctionnant grâce à la rétro-propagation de ce dernier. Dans notre cas, la fonction de coût à minimiser est l'entropie croisée, soit l'opposé du logarithme de la probabilité associée à la vraie classe. L'algorithme d'optimisation utilisé s'appelle Adadelta ZEILER [2012], il fonctionne bien pour le problème étudié ici et ne nécessite pas de réglage manuel de taux d'apprentissage initial ou autre paramètre.

2.5 Expériences et résultats

2.5.1 Hyper-paramètres

Les trois différents classifieurs proposés ont, en plus de leurs paramètres entraînables, quelques hyper-paramètres qui doivent être réglés manuellement qui sont : le choix de la couche d'agrégation (MaxPooling ou LSTM), le nombre de filtres de convolution dans les CNNs, la taille de leurs noyaux et, pour le MCNN seulement, la taille de la première couche de neurones et les échelles et fréquences de coupure des blocs de pré-traitement.

Concernant le FCN, les nombres de filtres de convolution des trois CNNs ont été fixés à 32, 64 et 32 et trois combinaisons de tailles de noyaux ont été testées, l'une avec un noyau plus long pour la couche du milieu et deux plus petits en

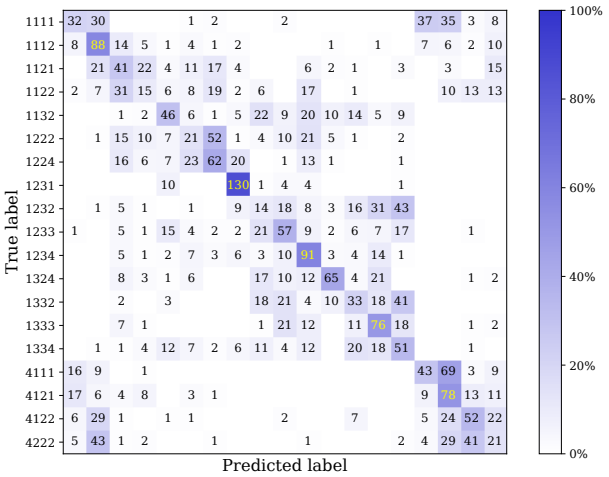


FIGURE 2.3 – Matrice de confusion du meilleur classifieur sur le jeu de données de test. Chaque case donne le nombre d'aéronef de la classe en ordonnée classée dans la classe en abscisse. Un classifieur parfait aurait une matrice de confusion diagonale.

première et troisième couche, une avec des tailles de noyaux croissantes et la dernière avec des tailles décroissantes.

Concernant le ResNet, le même genre de combinaisons a été testé au sein de chaque bloc. Le nombre de filtres de convolution a été fixé à 16 pour les couches du premier bloc, 32 pour le second et 64 pour le troisième.

Finalement, concernant le MCNN, beaucoup de combinaisons ont été testées, et seules les trois meilleures pour chacun des deux types d'agrégation sont présentés dans le tableau 2.2. Le paramètre n_c est le nombre de filtre de convolution dans les CNNs, le paramètre n_d est la taille de la couche de neurones munie d'une fonction d'activation ReLU, et n_f est le nombre d'échelles et de fréquences de coupure utilisées. La longueur des noyaux des convolutions a été fixée à 50 pour tous les CNNs. Une version du MCNN incluant des BNs a été entraînée pour chacune des configurations présentes dans le tableau.

2.5.2 Résultats principaux

Le tableau 2.2 résume les combinaisons d'hyper-paramètres testées pour les différents classifieurs, ainsi que leurs nombres de paramètres entraînables et leur précision sur le jeu de test après entraînement. On peut y constater que tous les algorithmes étudiés classifient entre 30% et 40% des cibles dans la bonne catégorie. Le meilleur classifieur d'après cette métrique est un MCNN muni de BNs avec une précision de 39.86%. Il est à noter que le gain de performance obtenu grâce à l'utilisation des BNs est en accord avec les conclusions de FAWAZ et collab. [2019].

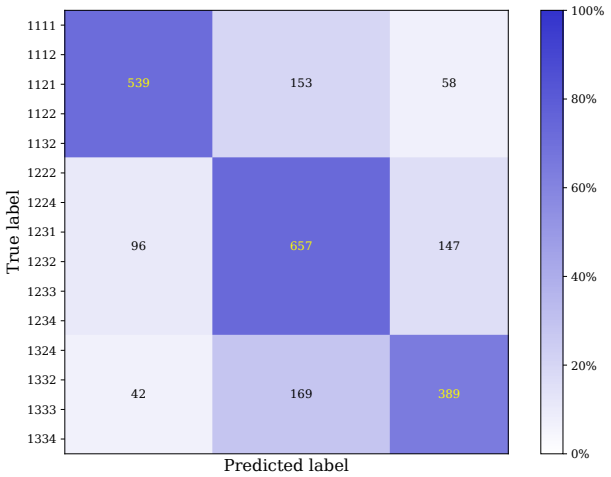


FIGURE 2.4 – Matrice de confusion du même classifieur que précédemment, adapté au problème restreint qu'est l'estimation de la WTC des avions.

2.5.3 Discussion

Le score de 39.86% de classifications correctes peut sembler faible, mais nous pensons qu'il ne l'est pas compte tenu du problème que nous cherchons à résoudre. En effet la granularité des labels exposés dans le tableau 2.1 est faible, et deux appareils appartenant à des classes distinctes peuvent très bien avoir des trajectoires presque identiques.

La matrice de confusion représentée dans la figure 2.3 donne une idée du type d'erreur qui peut arriver souvent et qui est même parfois inévitable. Par exemple, la classe 1231, généralement associée aux avions de chasse, est facilement reconnaissable, principalement en raison des fortes vélocité et manœuvrabilité des appareils qui la composent. Ainsi, cette classe atteint un taux de rappel de 87%. Cependant les petits avions monomoteur à piston (1111) sont souvent classifiés comme bimoteur (1112) ou comme hélicoptère léger à un moteur (4111 ou 4121) à la place en raison de la similarité de leurs cinématiques. La raison est que notre problème de classification est basé sur des classes issues des données ADS-B qui ne correspondent pas exactement à des catégories de trajectoires et sont d'ailleurs transmises aux contrôleurs aériens en tant qu'information complémentaire aux trajectoires estimées à partir des détections radar. En utilisant des classifieurs à l'état de l'art et une large gamme de descripteurs cinématiques des trajectoires, on obtient des résultats de classification relativement proches qui s'approchent vraisemblablement de la borne supérieure des résultats atteignables pour le problème considéré.

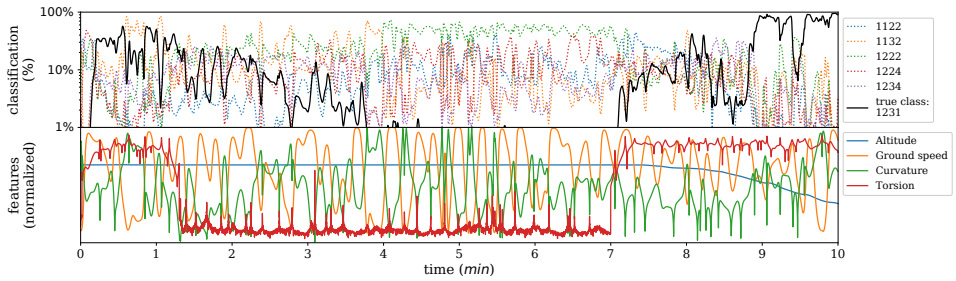


FIGURE 2.5 –
Courbes du haut : évolution en temps réel des probabilités attribuées aux classes les plus probables sur une échelle logarithmique. Courbes du bas : principaux descripteurs utilisés pour la classification normalisés et bornés par une fonction tanh.

2.5.4 Problèmes restreints de classification

En pratique, la granularité de la classification en 19 classes différentes issues des données ADS-B est probablement trop élevée pour les usages envisagés. Il est alors possible de regrouper les classes pour utiliser le classifieur entraîné dans la résolution de problèmes restreints. Par exemple, on pourrait se concentrer sur l'estimation de la taille d'un avion suivi en cherchant à connaître sa *Wake Turbulence Category* (WTC) correspondant à sa catégorie de poids. Une version légèrement modifiée du même classifieur permet alors d'obtenir une précision de 70.44% avec une matrice de confusion représentée en figure 2.4 pour ce problème à trois classes. Ce résultat est en effet bien plus satisfaisant. Les précisions de chacun des filtres sur ce problème restreint ainsi que sur celui de l'estimation de l'espèce de l'aéronef (avion ou hélicoptère) se trouve dans le tableau 2.2.

2.5.5 Qu'apprend le réseau ?

Pour compléter cette étude, observons la réaction du classifieur lorsqu'il reçoit un flux de données en temps réel afin de simuler un usage en conditions réelles de l'algorithme. Ici, il nous suffit de modifier notre couche d'agrégation pour qu'elle retourne la séquence entière au lieu du dernier point. Ainsi, le classifieur exprime à chaque temps la probabilité de l'appareil d'appartenir à chacune des classes en ne se basant que sur les données passées.

Nous proposons de nous pencher sur le cas particulier d'un appareil de classe 1231, soit un avion de chasse. La figure 2.5 montre l'évolution de la séquence au cours du temps pour une unique trajectoire. Elle donne une idée de ce qui a le plus d'impact sur le résultat en montrant ce qui coïncide avec avec des changements dans les probabilités. Le classifieur utilisé pour tracer ces courbes utilise un LSTM comme couche d'agrégation ce qui donne des probabilités plus fluctuantes que pour un Max.

Nous pouvons observer que la torsion joue un rôle important pour détecter les phases de manœuvre (et donc aider à classifier la cible). Il est intéressant

de constater que la courbure qui devrait être suffisante pour discriminer les périodes de manœuvre de celles de ligne droite a un comportement bien plus irrégulier. C'est cohérent avec le fait que la torsion avait déjà été identifiée comme une donnée importante pour caractériser les manœuvres des avions de chasse dans NABAA et BISHOP [2000], sur la base de 26 trajectoires réelles d'appareils militaires. Il est par contre intéressant de noter que dans notre cas, cette observation a été automatiquement "apprise" par l'algorithme.

On peut observer que la courte manœuvre durant la première minute aide l'algorithme à identifier correctement l'appareil de classe 1231 (avion de taille moyenne à un moteur à réaction, dans notre cas un avion de chasse) mise à part quelques hésitations avec la classe 1132 (bimoteur à réaction léger). Ensuite lorsque l'appareil vole en ligne droite, l'algorithme perd de sa certitude puisque les lignes droites ne sont pas discriminantes. On peut en déduire que la courte manœuvre du début n'était visiblement pas suffisante pour trancher définitivement la question. Cependant, sept minutes plus tard, l'appareil entame une descente et la probabilité de la vraie classe, à savoir 1231, remonte très rapidement jusqu'à avoisiner les 100%.

2.6 Conclusion

Différents réseaux de neurones ont été entraînés à résoudre un problème difficile de classification d'aéronefs à 19 classes basée uniquement sur des grandeurs cinématiques. Les architectures des classifieurs ainsi que le choix des grandeurs données en entrées garantissent que l'algorithme fonctionne quelque soit la longueur de la trajectoire (après un certain minimum de points) et que ses résultats soient invariants par rotation et translation dans le plan horizontal.

Après entraînement, le meilleur classifieur obtient une précision de près de 40% sur le jeu de données de test pour le problème de classification à 19 classes. Les classifieurs peuvent ensuite être adaptés pour résoudre des problèmes réduits sans qu'il soit nécessaire de repasser par une phase coûteuse d'entraînement en simplement supprimant ou fusionnant des classes. Par exemple, ce même classifieur obtient une précision de près de 74% lorsqu'il s'agit de déterminer la WTC des avions, ce qui est un résultat encourageant compte tenu du fait que le jeu de données utilisé est particulièrement bruité. La WTC est une information importante pour caractériser la taille et la puissance des aéronefs. De plus, les résultats peuvent être observées au cours du temps, il est ainsi possible d'inférer quels comportement dans la trajectoire ont pu causer les changements dans les probabilités exprimées et donc d'identifier les grandeurs les plus discriminantes dans notre jeu de données.

Au delà de la classification des appareils étudiée ici, nous pensons que les algorithmes et grandeurs utilisées ici peuvent aussi servir de base pour la résolution d'autres problèmes comme la détection de comportements agressifs ou

toute autre activité anormale par exemple.

2.7 Références

CHEN, S., H. WANG, F. XU et Y.-Q. JIN. 2016, «Target classification using the deep convolutional networks for sar images», *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, nº 8, p. 4806–4817. 22

CHOPIN, P., F. BARBARESCO et S. JOUABER. 2020-10-14, «Device for identifying a type of aircraft, associated identification method and computer program», . 24

CUI, Z., W. CHEN et Y. CHEN. 2016, «Multi-scale convolutional neural networks for time series classification», *arXiv preprint arXiv :1603.06995*. 28

DIZAJI, R. et H. GHADAKI. 2006, «Classification system for radar and sonar applications», . 22

FAWAZ, H. I., G. FORESTIER, J. WEBER, L. IDOUMGHAR et P.-A. MULLER. 2019, «Deep learning for time series classification : a review», *Data Mining and Knowledge Discovery*, vol. 33, nº 4, p. 917–963. 23, 28, 31

GHADAKI, H. et R. DIZAJI. 2006, «Target track classification for airport surveillance radar (asr)», dans *2006 IEEE Conference on Radar*, IEEE, p. 4–pp. 22

HE, K., X. ZHANG, S. REN et J. SUN. 2016, «Deep residual learning for image recognition», dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770–778. 29

IOFFE, S. et C. SZEGEDY. 2015, «Batch normalization : Accelerating deep network training by reducing internal covariate shift», *arXiv preprint arXiv :1502.03167*. 29

JEONG, Y.-S., M. K. JEONG et O. A. OMITAOMU. 2011, «Weighted dynamic time warping for time series classification», *Pattern recognition*, vol. 44, nº 9, p. 2231–2240. 27

LE GUENNEC, A., S. MALINOWSKI et R. TAVENARD. 2016, «Data augmentation for time series classification using convolutional neural networks», . 28

LEI, J. et C. LU. 2005, «Target classification based on micro-doppler signatures», dans *IEEE International Radar Conference, 2005.*, IEEE, p. 179–183. 22

MOHAJERIN, N., J. HISTON, R. DIZAJI et S. L. WASLANDER. 2014, «Feature extraction and radar track classification for detecting uavs in civillian airspace», dans *2014 IEEE Radar Conference*, IEEE, p. 0674–0679. 23

MOLCHANOV, P., K. EGIAZARIAN, J. ASTOLA, A. TOTSKY, S. LESHCHENKO et M. P. JARABO-AMORES. 2014, «Classification of aircraft using micro-doppler bicoherence-based features», *IEEE Transactions on Aerospace and Electronic systems*, vol. 50, nº 2, p. 1455–1467. 22

NABAA, N. et R. H. BISHOP. 2000, «Validation and comparison of coordinated turn aircraft maneuver models», *IEEE Transactions on aerospace and electronic systems*, vol. 36, nº 1, p. 250–259. 23, 34

NASRABADI, N. M. 2019, «Deeptarget : An automatic target recognition using deep convolutional neural networks», *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, nº 6, p. 2687–2697. 22

PIERUCCI, L. et L. BOCCHI. 2007, «Improvements of radar clutter classification in air traffic control environment», dans *2007 IEEE International Symposium on Signal Processing and Information Technology*, IEEE, p. 721–724. 22

RATANAMAHATANA, C. A. et E. KEOGH. 2004, «Making time-series classification more accurate using learned constraints», dans *Proceedings of the 2004 SIAM international conference on data mining*, SIAM, p. 11–22. 27

SERRÀ, J., S. PASCUAL et A. KARATZOGLOU. 2018, «Towards a universal neural network encoder for time series.», dans *CCIA*, p. 120–129. 28

TANISARO, P. et G. HEIDEMANN. 2016, «Time series classification using time warping invariant echo state networks», dans *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, p. 831–836. 28

THIAGARAJAN, J. J., K. N. RAMAMURTHY, P. KNEE, A. SPANIAS et V. BERISHA. 2010, «Sparse representations for automatic target classification in sar images», dans *2010 4th international symposium on communications, control and signal processing (ISCCSP)*, IEEE, p. 1–4. 22

WANG, Z., W. YAN et T. OATES. 2017, «Time series classification from scratch with deep neural networks : A strong baseline», dans *2017 International joint conference on neural networks (IJCNN)*, IEEE, p. 1578–1585. 23, 27, 28

XI, X., E. KEOGH, C. SHELTON, L. WEI et C. A. RATANAMAHATANA. 2006, «Fast time series classification using numerosity reduction», dans *Proceedings of the 23rd international conference on Machine learning*, p. 1033–1040. 27

YAN, W., Z. ZHU et R. HU. 1997, «A hybrid genetic/bp algorithm and its application for radar target classification», dans *Proceedings of the IEEE 1997 National Aerospace and Electronics Conference. NAECON 1997*, vol. 2, IEEE, p. 981–984. 22

ZEILER, M. D. 2012, «Adadelta : an adaptive learning rate method», *arXiv preprint arXiv :1212.5701*. 30

ZHAO, B., H. LU, S. CHEN, J. LIU et D. WU. 2017, «Convolutional neural networks for time series classification», *Journal of Systems Engineering and Electronics*, vol. 28, nº 1, p. 162–169. 28

ZHENG, Y., Q. LIU, E. CHEN, Y. GE et J. L. ZHAO. 2014, «Time series classification using multi-channels deep convolutional neural networks», dans *International Conference on Web-Age Information Management*, Springer, p. 298–310. 28

ZYWECK, A. et R. E. BOGNER. 1996, «Radar target classification of commercial aircraft», *IEEE Transactions on Aerospace and Electronic systems*, vol. 32, nº 2, p. 598–606. 22

Chapitre 3

Amélioration des filtres basés sur le modèle MRU, filtrage adaptatif

Sommaire

3.1 Résumé

Ce chapitre propose une approche par réseaux de neurones des filtres de Kalman adaptatifs qui utilisent un unique modèle cinématique mais dont les paramètres peuvent être ajustés dynamiquement selon les besoins. Cette adaptation que nous appellerons par la suite NNAKF permet pour un moindre de coût en termes de temps de calculs d'améliorer la précision d'un filtre de Kalman de façon significative. Elle peut être décomposée en deux parties, un ensemble de matrices de bruit entièrement paramétrables qui permettront d'ajuster la confiance que le filtre a envers la prédiction donnée par le modèle, et un réseau de neurones qui fera office de stratégie d'adaptation en activant ou non chacune des matrices à sa disposition. Après avoir exploré les différentes possibilités d'hyper-paramétrage du NNAKF, ses performances en termes de précision sont favorablement comparées aux filtres de la littérature. Ensuite, une analyse plus poussée du filtre sur des scénarios spécifiques permet de mettre en lumière le fonctionnement et la robustesse de celui-ci.

3.2 Introduction

Ce chapitre traite de l'utilisation de filtres de Kalman adaptatifs. Il s'agit de variations du filtre de Kalman classique pour lesquelles les paramètres, à savoir les bruits de modèles, sont ajustés dynamiquement au cours du pistage. Sauf exceptions, les filtres présentés tout au long de ce chapitre utiliseront pour modèle cinématique le Mouvement Rectiligne Uniforme (MRU) pour leur étape de prédiction. Il s'agit d'un modèle très simple dont les performances limitées laissent une large place à l'amélioration, ce qui nous permettra de mieux comparer les filtres étudiés. Ce modèle est plus précisément décrit ci-dessous.

Définition formelle du modèle MRU Dans un modèle MRU, seules la position et la vitesse de la cible suffisent à prédire le mouvement. L'état de la cible réunit donc ces informations comme dans l'équation 3.1. Les coefficients du système d'équations différentielles stochastiques sont représentés par la matrice bloc de l'équation 3.2 où I_3 est la matrice identité en dimension 3 et $0_{3,3}$ la matrice nulle de dimension $(3,3)$, on en déduit facilement l'opérateur d'évolution présenté dans l'équation 3.3. En découpant la matrice de bruit en blocs comme dans l'équation 3.4, on peut l'intégrer pour obtenir le bruit de modèle de l'équation 3.5.

$$\chi = \begin{pmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{pmatrix} \tag{3.1}$$

$$A = \begin{pmatrix} 0_{3,3} & I_3 \\ 0_{3,3} & 0_{3,3} \end{pmatrix} \tag{3.2}$$

$$F(\Delta t) = \begin{pmatrix} I_3 & \Delta t I_3 \\ 0_{3,3} & I_3 \end{pmatrix} \tag{3.3}$$

$$Q = \begin{pmatrix} Q_P & C_{P,P'} \\ C_{P,P'}^T & Q_{P'} \end{pmatrix} \tag{3.4}$$

$$Q_{\Delta t} = \begin{pmatrix} \Delta t Q_P + \frac{1}{2}\Delta t^2 (C_{P,P'} + C_{P,P'}^T) + \frac{1}{3}\Delta t^3 Q_{P'} & \Delta t C_{P,P'} + \frac{1}{2}\Delta t^2 Q_{P'} \\ \Delta t C_{P,P'}^T + \frac{1}{2}\Delta t^2 Q_{P'} & \Delta t Q_{P'} \end{pmatrix} \tag{3.5}$$

3.2.1 Influence du bruit de modèle

Lorsque le modèle cinématique du filtre de Kalman est fixé, le seul réglage restant pour définir le modèle dynamique est l'amplitude du bruit de modèle. La Figure 3.1 illustre l'influence de ce paramètre sur les résultats du filtre. On y constate que lorsque le bruit est bas, et donc que la confiance en le modèle est haute, le filtrage est très précis lorsque la trajectoire respecte bien la cinématique attendue, à savoir la ligne droite lors de la première partie du tracé. En revanche lorsque l'on sort de ce cadre, comme lors du virage présent dans la deuxième partie, le filtre a beaucoup de mal à suivre. Par contre avec un niveau de bruit élevé, le filtre fait plus confiance aux mesures qu'à ses prédictions et a de meilleurs résultats sur la partie manœuvrante, mais la précision est alors limitée par le bruit de mesure et l'estimation du vecteur vitesse dans la partie ligne droite est dégradée car le filtre accorde une importance moindre au modèle ligne droite.

Une idée assez naturelle serait alors de chercher à détecter les manœuvres afin d'ajuster le bruit de modèle en conséquence et de profiter du meilleur des deux mondes [1]. C'est l'idée du filtrage adaptatif et plus particulièrement ici de l'adaptation de CASTELLA [1980].

1. Une autre possibilité serait d'utiliser deux filtres en parallèle et de pondérer leurs estimations en fonction de leurs vraisemblances, on parle alors de filtre à modèles multiples, et ce sera l'objet du chapitre suivant.

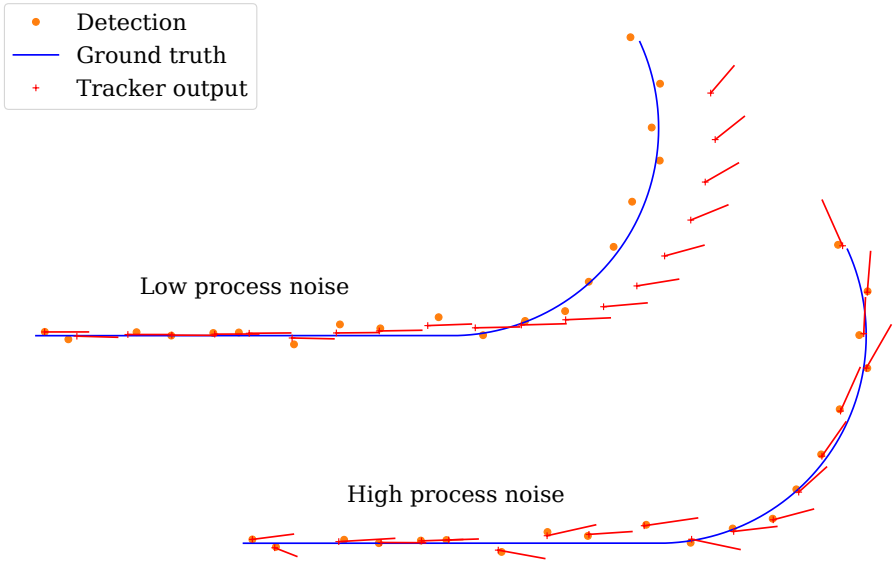


FIGURE 3.1 – Exemples de filtrage d'un même scénario 2D par deux filtres de Kalman basés sur un modèle MRU avec des réglages de bruit de modèle différents. La courbe bleue est la trajectoire réelle à retrouver, les points oranges sont les détections fournies aux filtres pour cette tâche et en rouge sont représentées les estimations qui en résultent sous la forme d'une croix pour la position et d'un trait représentant le vecteur vitesse.

3.2.2 Adaptation de Castella

L'adaptation de Castella, dont un schéma est représenté Figure 3.2, fonctionne en se basant sur l'innovation normalisée entre la prédiction et la mesure sur chacun des trois axes de l'espace. La formule de cette innovation est explicitée Équation 3.6. Le bruit de modèle sur la vitesse pour chaque axe correspondant est alors ajusté selon une fonction affine par morceaux comme représentée Figure 3.2. L'adaptation ne requiert alors que l'ajout de quelques valeurs seuils de bruit et d'innovation à configurer et reste relativement simple à régler manuellement. La version de l'adaptation de Castella implémentée, entraînée et testée ici utilise une matrice de bruit fixe Q entièrement paramétrable pour la prédiction comme pour un filtre classique et un bruit additif $q^{+}_{\mathrm{V},n}$ sur les vitesses dérivant de l'innovation normalisée selon une fonction sigmoïde comme décrite en Équation 3.7. Les paramètres $p_{i,1}$ gèrent le seuil maximal d'adaptation de chaque axe, et les paramètres $p_{i,2}$ et $p_{i,3}$ remplacent les seuils d'innovation. Ils sont tous réglés automatiquement lors de l'apprentissage. Les résultats du filtre de Kalman avec et sans adaptation pourront être retrouvés en Table 3.3 sous forme de racine de l'erreur quadratique moyenne (REQM).

$$(\mathrm{I}_n)_i = \frac{(\mathrm{X}_{n|n-1} - \mathrm{Y}_n)_i^2}{(\mathrm{P}_{n|n-1})_{i,i} + (\mathrm{R}_n)_{i,i}} \tag{3.6}$$

$$(q^+_{\mathrm{V},n})_i = p_{i,1}\sigma(p_{i,2}(\mathrm{I}_n)_i + p_{i,3}) \tag{3.7}$$

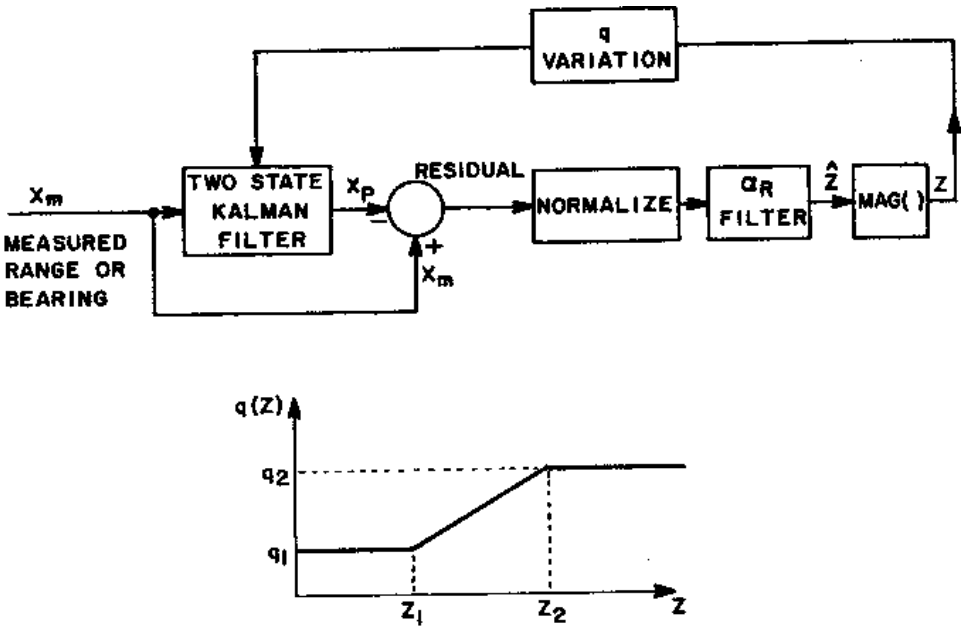


FIGURE 3.2 – Illustration extraite de CASTELLA [1980] illustrant le fonctionnement de son adaptation.

3.2.3 Contributions et organisation du chapitre

Notre approche du filtre adaptatif, qui sera présentée dans la section 3.3, s'inspire de l'adaptation de Castella et en étend les capacités à la fois dans la forme des bruits additifs puisque la totalité des termes de variances et covariances des matrices seront utilisés, et dans leur utilisation qui sera opérée par un réseau de neurones. Au cours de la section 3.4, nous nous pencherons sur le choix des hyper-paramètres de notre filtre (3.4.1), ses performances par rapport aux autres exemples de la littérature (3.4.2) et sa robustesse face à des trajectoires différents de celles rencontrées lors de la phase d'entraînement (3.4.4).

3.3 NNAKF

3.3.1 Fonctionnement

Le NNAKF, pour Neural Network Adaptive Kalman Filter, introduit dans JOUABER et collab. [2021] et représenté Figure 3.3 est un filtre de Kalman adap-

tatif faisant intervenir un réseau de neurones. Pour fonctionner, un nombre N de matrices de bruit de modèles $(Q_i)_{1\leq i\leq N}$ sont apprises en plus de celle de l'étape de prédiction classique Q_0. En parallèle, un réseau de neurones utilisant les mêmes innovations normalisées que le filtre adaptatif de Castella en entrée est entraîné à sélectionner les matrices apprises pour les utiliser comme bruit additif. Pour ce faire, il exprimera N coefficients $(\sigma_i)_{1\leq i\leq N}$ bornés entre 0 et 1 grâce à l'utilisation d'une fonction d'activation finale en sigmoïde. Le bruit additif sera alors la somme des matrices de bruit pondérées par les coefficients. La valeur de la matrice de bruit totale est exprimée en équation 3.8. On peut voir l'ensemble de matrices apprises comme une boîte à outils dont le réseau de neurones pourra se servir pour gérer l'adaptation. Les matrices de bruit ainsi que les paramètres du réseau de neurones sont optimisés conjointement grâce à un algorithme de descente de gradient.

$$Q = Q_0 + \sum_{i=1}^{N} \sigma_i Q_i \tag{3.8}$$

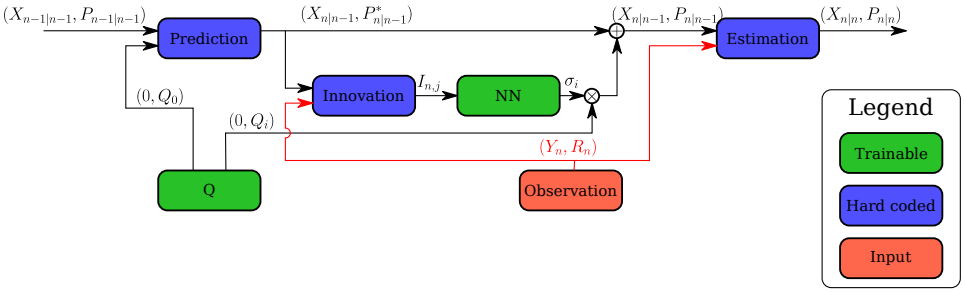


FIGURE 3.3 – Schéma du NNAKF.

3.3.2 Motivations

Le choix de cette architecture a été motivé par la volonté d'obtenir un filtre à la fois fiable et peu économe en calculs. En effet, le fonctionnement du NNAKF reste assez proche de celui de l'adaptation de Castella qui en est même un cas limite lorsque le réseau de neurones est constitué d'une simple couche et que $N \geq 3$. De plus, les calculs supplémentaires dûs à l'utilisation du NNAKF sont relativement peu conséquents, l'étape d'estimation nécessitant une inversion de matrice n'est toujours effectuée qu'une fois par cycle. Un autre avantage du NNAKF sur l'adaptation de Castella est la possibilité d'élargir ses capacités au besoin selon le problème à résoudre en jouant sur l'architecture du réseau de neurones et le nombre de matrices de bruit d'adaptation. Notamment l'utilisation d'une cellule LSTM permet au filtre de garder en mémoire des informations sur le passé de la cible pour une adaptation plus adéquate.

3.4 Expériences numériques

Dans cette section nous chercherons dans un premier temps à configurer au mieux le NNAKF pour le problème considéré, à savoir le filtrage sur le jeu de données simulé, avant de le comparer à d'autres filtres de la littérature et d'en étudier le fonctionnement. Tous les filtres considérés ici sont réglés automatiquement grâce aux jeux de données d'entraînement et de validation définis en annexe A.1, à la fonction de coût définie en annexe A.2 et à l'algorithme d'optimisation Adadelta ZEILER [2012]. Pour chaque filtre, l'entraînement a été effectué cinq fois et seule la configuration présentant le coût le plus bas sur le jeu de validation est conservée pour être évaluée et comparée sur le jeu de test.

3.4.1 Hyper-paramétrage du NNAKF pour le problème consiréré

Choix du NN

Le premier choix conséquent pour obtenir le NNAKF le plus adéquat au problème considéré est celui de l'architecture du réseau de neurones régissant l'adaptation. Les données fournies au réseau étant assez pauvres (trois valeurs), nous n'utiliserons que des architectures simples, à savoir une simple couche de neurones, un MLP à deux couches ou une cellule LSTM. Le nombre de matrices de bruit est fixé à trois pour rester proche du fonctionnement de l'adaptation de Castella, en effet il s'agit d'un cas limite du NNAKF utilisant une simple couche de neurones. Dans le cas du LSTM, les valeurs initiales de la cellule (h_0 et c_0) sont identiques pour toutes les trajectoires et automatiquement réglées par l'apprentissage.

Les mesures d'imprécision sur différentes métriques de ces trois filtres sont données sous forme de racines d'erreurs quadratiques moyennes (REQM) dans la table 3.1. On peut y remarquer en comparant les deux premières lignes que l'ajout d'une seconde couche de neurones n'apporte rien aux performances, on pourrait même y déceler une très légère détérioration. En revanche l'utilisation de la cellule LSTM permet d'obtenir les meilleurs résultats sur trois des quatre métriques, et l'écart est même relativement conséquent ($> 5\%$) pour la position et le cap.

Cette différence de précision peut probablement s'expliquer par la mémoire de la cellule LSTM qui lui permet plus facilement de différencier les manœuvres des points aberrants. Nous faisons donc le choix pour la suite de continuer à travailler avec une cellule LSTM comme réseau de neurones pour le NNAKF.

Choix du nombre de matrices

À présent on peut s'intéresser au nombre de matrices dont disposera le NNAKF. On peut le voir comme un réglage fin de la complexité de l'adaptation. Plus ce nombre est élevé plus le filtre aura de liberté pour établir sa stratégie d'adaptation, ce qui signifie généralement de meilleurs résultats, mais une

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
FC1	54.32	46.69	**3.33**	13.56	17.348
FC2	54.65	46.68	3.35	13.69	17.377
LSTM	**51.14**	**46.65**	3.48	**12.46**	**16.924**

TABLEAU 3.1 – REQM sur différentes métriques de filtres de Kalman utilisant une adaptation NNAKF à trois matrices et différentes architectures de réseaux de neurones : une couche (FC1), deux couches (FC2) ou une cellule LSTM. Dans le cas des deux couches, la taille de la première est fixée au double de celle de la seconde, en l'occurence 6. La fonction de coût obtenue pour chaque filtre est aussi représentée dans la dernière colonne. Le meilleur résultat pour chaque métrique est inscrit en gras.

phase d'entraînement plus longue et un fonctionnement plus difficile à interpréter. On considérera dans cette expérience des NNAKF encore basés sur un modèle MRU et utilisant une cellule LSTM comme réseau de neurones. Les valeurs prises par l'hyper-paramètre qui régit le nombre de matrices d'adaptation évoluera de 1 à 10.

La figure 3.4 représente les valeurs de la fonction de coût obtenues par les filtres après apprentissage en fonction du nombre de matrices utilisées par le NNAKF. On remarque une forte amélioration lorsque l'on passe de trois à quatre matrices. En dehors de cette chute, la fonction de coût décroît relativement lentement. Les valeurs de REQM en fonction de l'hyper-paramètre sont représentées figure 3.5. L'incidence sur la précision en position, en vitesse et en cap semble quasiment nulle. En revanche l'évolution de la REQM en altitude semble suivre le même profil que celle de la fonction de coût. On peut en déduire que c'est la précision sur l'altitude qui est à l'origine de la chute de la fonction de coût.

L'existence d'une matrice de bruit spécialisée sur l'axe vertical dans le NNAKF à 4 matrices non présente dans celui à 3 matrices ni dans celui à 4 matrices ayant obtenu la valeur de fonction de coût la plus haute est probable, cette possibilité sera étudiée plus en détail dans l'expérience 3.4.3. Nous continuerons dans la suite à utiliser 10 matrices d'adaptation pour le NNAKF.

Comparaison à d'autres modèles

Dans cette troisième expérience, on s'intéresse à l'importance du choix du modèle cinématique du filtre de Kalman. Nous introduisons deux nouveaux modèles dont les définitions formelles peuvent être trouvées en annexe. Le premier est le Mouvement uniformément accéléré (MUA) défini en annexe A.3.2, il nécessite d'étendre l'espace des états défini pour le modèle MRU en y ajoutant les valeurs d'accélération sur les trois axes. Le second est le modèle de SINGER [1970] défini en annexe A.3.3 qui nécessite aussi le même espace d'état que le modèle MUA, les valeurs de l'accélération ne ici pas constantes mais décroissent exponentiellement selon un paramètre $\alpha > 0$ fixe qui sera automatiquement ré-

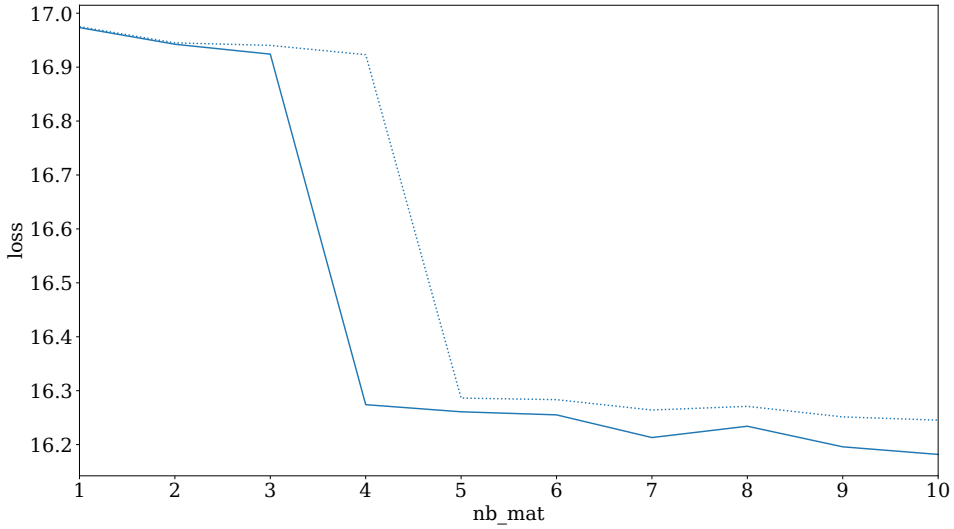


FIGURE 3.4 – Valeurs de la fonction de coût sur le jeu de test obtenues par le NNAKF basé sur un modèle MRU avec cellule LSTM comme réseau de neurones en fonction du nombre de matrices d'adaptation. La ligne pleine représente la valeur minimale obtenue parmi les cinq entraînements effectués, et donc celle du filtre utilisé par la suite pour le calcul des REQM. La ligne en pointillés représente la valeur maximale.

glé au cours de l'apprentissage. Pour chacun de ces deux modèles, l'accélération initiale est fixée à 0 et les variances correspondantes sont identiques pour toutes les trajectoires et automatiquement réglées à l'apprentissage, les termes de covariances initiales liés aux accélérations sont nuls. Les définitions formelles de ces deux modèles sont disponibles en annexe à la fin de ce chapitre.

Les REQM obtenues avec les différents modèles en utilisant à chaque fois un NNAKF à dix matrices et une cellule LSTM sont présentées dans la table 3.2. Les résultats sont relativement proches, on peut toutefois noter certains détails. Tout d'abord, le modèle MUA obtient les moins bons résultats dans toutes les métriques, il s'agit d'un modèle qui n'est que très rarement utilisé dans le domaine du pistage radar. Ensuite, le modèle de Singer obtient le meilleur score sur trois des quatre métriques ainsi que la fonction de coût la plus basse. Ce résultat est prévisible étant donné que les modèles MRU et MUA en sont des cas limites selon si le paramètre α est très grand ou très petit.

3.4.2 Comparaison à d'autres filtres

Dans cette section nous cherchons à évaluer le gain de précision obtenu en utilisant le NNAKF précédemment configuré et à le comparer à d'autres filtres et adaptations de la littérature. Le paramétrage de chacun des algorithmes sera fait automatiquement et dans les mêmes conditions que dans la section précédente.

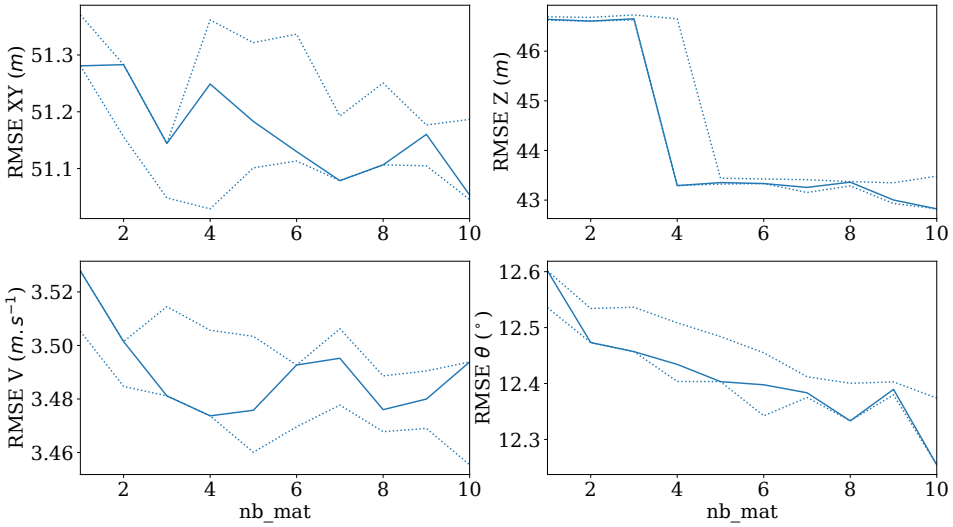


FIGURE 3.5 – Évolution des REQM en position (XY), altitude (Z), vitesse (V) et cap (θ) en fonction du nombre de matrices d'adaptation. La ligne pleine représente les valeurs obtenues pour les filtres ayant obtenus la plus faible valeur sur la fonction de coût au cours de cinq entraînements. Les lignes en pointillés représentent des valeurs minimales et maximales obtenues sur les métriques.

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
MRU	51.05	42.82	**3.49**	12.25	16.182
MUA	51.19	43.67	4.22	12.55	16.267
Singer	**50.78**	**41.60**	3.57	**12.14**	**15.858**

TABLEAU 3.2 – REQM sur différentes métriques de filtres de Kalman utilisant une adaptation NNAKF à dix matrices et une cellule LSTM en guise de réseau de neurones mais avec trois modèles cinématiques différents. La fonction de coût obtenue pour chaque filtre est aussi représentée dans la dernière colonne. Le meilleur résultat pour chaque métrique est inscrit en gras.

Bornes inférieures

Afin de comparer les différents filtres et plus particulièrement les améliorations apportées par les différentes adaptations, les résultats de deux filtres idéaux (mais pas réalistes) sont étudiés :

— le premier nommé Oracle, est un EKF qui utilise les commandes de vol et le modèle cinématique qui ont servi à générer les trajectoires et un bruit de modèle nul. L'erreur commise par l'Oracle est uniquement due au bruit des mesures et peut être considérée comme une borne inférieure pour le problème considéré.

— le second, plus adapté pour comparer les filtres présentés dans ce chapitre, a été nommé PbPO pour Point-by-Point Optimization. Il s'agit d'un

KF utilisant un modèle MRU pour lequel on a optimisé le bruit de modèle à chaque étape d'estimation afin de minimiser la fonction de coût. Ainsi, chaque itération du filtre de Kalman nécessite la résolution d'un problème d'optimisation complexe puisqu'il s'agit de trouver la paramétrisation θ de la matrice Q qui minimise la fonction H donnée dans l'équation 3.10. Faute de solution analytique, l'algorithme de descente de gradient Adadelta ZEILER [2012] a été utilisé grâce à la formule simplifiée du gradient exprimée dans l'équation 3.11. Enfin la matrice Q a été paramétrée comme le produit d'une matrice W entièrement paramétrable par sa transposée ($Q = W.W^T$). Les paramètres θ sont donc les coefficients W_{ij} et le gradient de la matrice Q par ceux-ci sont exprimés dans l'équation 3.12. L'erreur commise par le PbPO peut être vue comme une borne inférieure pour le problème considéré lorsque l'on se restreint aux filtres basés sur un modèle MRU et utilisant une stratégie d'adaptation du bruit de modèle comme Castella ou le NNAKF.

$$\begin{aligned} K_k(Q) &= (P_{k|k-1} + Q(\theta))H^T(P_{k|k-1} + Q(\theta) + R_k)^{-1} \\ \Delta X_k^r &= X_{k|k-1} - Y_{true} \\ \Delta X_k^m &= X_{k|k-1} - Y_{meas} \end{aligned} \tag{3.9}$$

$$\begin{aligned} H = &-(\Delta X_k^r - K_k \Delta X_k^m)^T (P_{k|k-1} + Q(\theta))^{-1} (I_6 - K_k H)^{-1} (\Delta X_k^r - K_k \Delta X_k^m) \\ &+ \ln|I_6 - K_k H| + \ln|(P_{k|k-1} + Q(\theta))| \end{aligned} \tag{3.10}$$

$$\begin{aligned} \frac{\partial H}{\partial \theta} = &(\Delta X_k^r - K_k \Delta X_k^m)^T (P_{k|k-1} + Q(\theta))^{-1} \frac{\partial Q}{\partial \theta} (P_{k|k-1} + Q(\theta))^{-1} (\Delta X_k^r + K_k \Delta X_k^m) \\ &+ Tr((P_{k|k-1} + Q(\theta))^{-1} (I_6 - K_k H) \frac{\partial Q}{\partial \theta}) \end{aligned} \tag{3.11}$$

$$\frac{\partial Q_{ab}}{\partial W_{ij}} = \delta_{ai} W_{bj} + \delta_{bi} W_{aj} \tag{3.12}$$

Filtres de la littérature

Parmi les filtres de la littérature que nous avons entraînés et comparés, deux ont déjà été introduits :

— le filtre de Kalman simple doté d'un modèle cinématique MRU (Section 3.2)

— l'adaptation de Castella (Section 3.2.2)

Les deux autres entraînés pour la comparaison sont :

— l'adaptation de VAIDEHI et collab. [2001] qui utilise aussi un modèle MRU. L'adaptation consiste à utiliser un MLP à trois couches qui prend en entrée

un certain nombre de données pour exprimer une correction à ajouter à la position dans l'état après estimation. Les données en entrée sont l'innovation (différence entre position mesurée et position prédite), la diagonale du bloc (3×3) du gain lié à la position, et la correction en position utilisée à l'étape d'estimation. Les dimensions des couches de neurones dans l'ordre sont 8, 5 et 3.

— une version allégée du LSTM-KF évoqué dans la section 1.3.3 et proposé par Coskun et collab. [2017]. Dans cette version, la partie boite noire ainsi que le LSTM définissant la matrice de bruit de mesure ont été retirés puisque les informations qu'elles sont censées déterminer sont déjà données dans notre problème. Il s'agit donc d'un EKF dans lequel l'étape de prédiction est régie par deux réseaux de neurones tous deux constitués d'une cellule LSTM suivie d'une couche de neurones sans fonction d'activation. Le premier réseau donne le terme à ajouter à la précédente estimation de l'état pour obtenir l'espérance de l'état prédit en se basant sur cette estimation. Le second réseau utilise l'état prédit issu du premier réseau pour définir la matrice de bruit de modèle, ce bruit ne sera pas intégré comme pour les autres filtre mais utilisé tel quel. Il est à noter que le pas de temps n'est pas pris en compte dans l'étape de prédiction et ce filtre ne peut donc fonctionner correctement que sur des jeux de données avec un pas de temps régulier et identique entre les trajectoires.

Résultats

Les mesures de REQM exprimées comme précédemment sont retranscrites dans la table 3.3. Les deux premières lignes montrent les deux bornes inférieures, les quatre suivantes sont pour les filtres de la littérature, et la dernière pour le NNAKF à dix matrices et une cellule LSTM.

Dans la table 3.4, nous mettons de côté les filtres de la littérature à réseaux de neurones (Vaidehi et LSTM-KF) ainsi que l'Oracle pour se concentrer sur les filtres classiques et le NNAKF et leur relation à la borne PbPO. On représente dans cette table les RMS des écarts d'erreur entre les filtres et la borne. Si la PbPO vise à simuler les résultats qu'on obtiendrait avec une stratégie d'adaptation du bruit de modèle idéale, cette mesure quantifie les erreurs évitables des différentes stratégies, ou de l'absence de stratégie dans le cas du KF.

Finalement la table 3.5 exprime la diminution relative de cette erreur évitable lorsque l'on ajoute une des deux stratégies d'adaptation du bruit de modèle (Castella et NNAKF). Un score de 0 signifie que l'adaptation n'a aucun effet et un score de 1 signifie que la borne PbPO a été atteinte.

Interprétations

Sur chacune des métriques, c'est le NNAKF qui obtient les meilleurs résultats (en mettant de côté l'Oracle et la PbPO bien entendu). Il est d'ailleurs à

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
Oracle	28.26	13.66	2.50	3.37	NA
PbPO	30.66	31.13	1.63	6.40	8.134
KF	59.62	46.69	4.77	17.51	18.467
Castella	54.11	46.62	3.81	14.29	17.507
Vaidehi	59.29	46.73	4.81	17.47	18.405
LSTM-KF	85.76	47.95	35.29	78.19	25.507
NNAKF	**51.05**	**42.82**	**3.49**	**12.25**	**16.182**

TABLEAU 3.3 – REQM sur différentes métriques des filtres. La fonction de coût obtenue pour chaque filtre est aussi représentée dans la dernière colonne. La vitesse verticale étant à la fois une commande de vol et une variable de l'état, elle est connue sans incertitude par "l'Oracle" dont la matrice de covariance n'est alors pas de rang maximal et la fonction de coût telle qu'elle est calculée normalement n'est pas définie. Le meilleur résultat pour chaque métrique est inscrit en gras.

Filter	Position (m)	Altitude (m)	Speed ($m.s^{-1}$)	Heading (°)
KF	37.47	28.70	4.05	14.84
Castella	30.73	28.41	3.04	11.28
NNAKF	**27.28**	**23.63**	**2.66**	**8.81**

TABLEAU 3.4 – REQM des différences d'erreur entre différents filtres et la borne PbPO (minorée à 0 dans les cas où les filtres font mieux que la borne). Il s'agit ici de quantifier l'erreur que l'on pourrait éviter si l'on trouvait une stratégie d'adaptation optimale.

noter qu'il s'agit du seul filtre capable d'améliorer la précision en altitude de façon significative. Ensuite, excepté sur l'altitude donc, l'adaptation de Castella est la seule autre à montrer une amélioration significative sur les différentes métriques. L'adaptation de Vaidehi a un effet très limité, tandis que les résultats du LSTM-KF sont franchement mauvais.

Les faibles performances de ce dernier peuvent avoir plusieurs explications. Tout d'abord il s'agit d'un EKF dont l'étape de prédiction est plutôt compliquée pour l'espérance et approximée par le jacobien pour la covariance, tout ceci rend le filtre assez instable et l'apprentissage "chaotique". Il pourrait être envisageable de lui donner un coup de pouce en ne lui faisant pas apprendre un modèle d'évolution à partir de rien mais plutôt une correction à appliquer à modèle prédéfini, comme le modèle MRU par exemple. Toutefois le problème des contraintes de pas de temps évoqué à la fin de la partie 3.4.2 reste entier et rend le filtre inadapté dans un contexte réaliste de pistage radar où le temps entre les détections consécutives est irrégulier. Nous n'irons donc pas plus loin dans l'étude de cette solution.

Filter	Position	Altitude	Speed	Heading
Castella	0.1799	0.0100	0.2498	0.2396
NNAKF	**0.2720**	**0.1765**	**0.3433**	**0.4059**

TABLEAU 3.5 – Écarts relatifs des mesures exprimées dans la table 3.4 par rapport au KF sans adaptation.

3.4.3 Matrices de bruit apprises

Filtre de Kalman

L'analyse des matrices de bruit apprises par nos filtres peut nous aider à en comprendre le fonctionnement. Dans le cas du KF simple, il s'agit même de l'unique réglage issu de l'apprentissage. La matrice est représentée en figure 3.6.

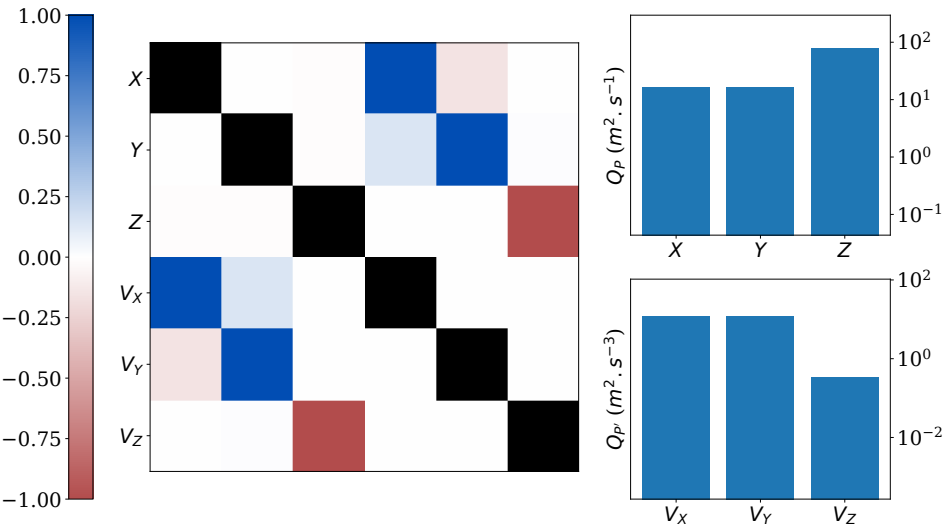


FIGURE 3.6 – Représentation de la matrice de bruit de modèle apprise pour le KF simple. La matrice sur la gauche représente les coefficients de corrélation entre les différentes variables de position et de vitesse. Les diagrammes à droites représentent les valeurs de variance de ces variables sur une échelle logarithmique. Les valeurs minimales et maximales de ces diagrammes seront les mêmes pour toutes les représentations de matrices de bruit de ce chapitre afin d'en faciliter la comparaison.

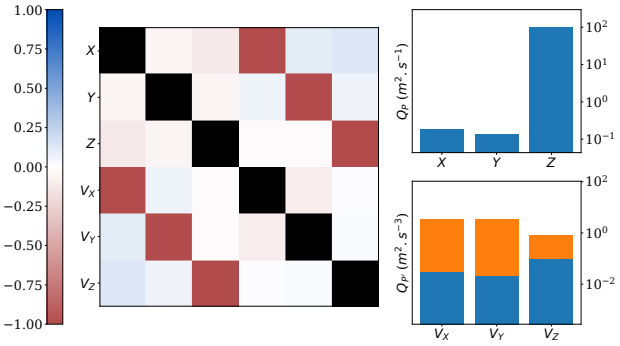
On peut y constater que le bruit agit de façon similaire sur les deux axes horizontaux, mais qu'il est différent sur l'axe vertical. Le bruit qui agit directement sur la position est plus fort pour l'axe vertical que pour les axes horizontaux, et inversement pour le bruit en vitesse. Cela reflète la façon dont les trajectoires ont été générées puisque le comportement des cibles n'est pas le même sur l'axe vertical que sur les axes horizontaux. La commande de vol qui agit sur l'altitude est la vitesse verticale, dérivée première de la position, alors que celles qui agissent sur les axes horizontaux sont l'accélération et le taux de virage, liés aux dérivées

secondes de la position.

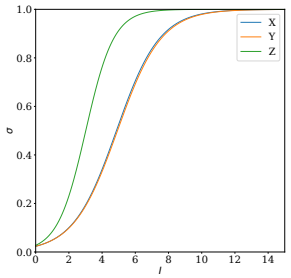
Autre remarque intéressante, la matrice apprise est creuse, et donc facile à interpréter, et utilise les termes non-diagonaux entre position et vitesse d'un même axe, ce que les ingénieurs "humains" ne font que très rarement voire jamais. Cette corrélation est positive pour les axes horizontaux et négative sur l'axe vertical ce qui après intégration, selon le pas de temps, peut compenser l'écart de bruit en position entre les axes. On peut en déduire que la principale différence entre vertical et horizontal pour le bruit de modèle réside dans la vitesse, le filtre sera plus enclin à mettre à jour son cap et sa vitesse au sol que sa vitesse verticale.

Adaptation de Castella

Passons maintenant à l'adaptation de Castella. En plus d'une matrice de bruit de base (représentée en figure 3.7a), les paramètres de l'adaptation doivent aussi être réglés afin de donner l'évolution du bruit de modèle supplémentaire en fonction du carré de l'innovation normalisé I. Les valeurs maximales de variances d'adaptation sont représentées figure 3.7a et les valeurs des fonctions sigmoïdes qui activent l'adaptation sont représentées sur la figure 3.7b.



(a) Représentation de la matrice de bruit de modèle apprise pour l'adaptation de Castella. En plus des coefficients de corrélation et des variances du bruit de base représentés comme sur la figure 3.6, les barres oranges des diagrammes de droite montrent l'amplitude de l'adaptation.



(b) Courbes des niveaux d'activation de l'adaptation de Castella en fonction du carré de l'innovation normalisé I sur les différents axes. Les courbes correspondant aux axes X et Y sont pratiquement confondues.

Excepté pour l'altitude, le bruit de modèle sans adaptation est bien plus faible que pour le KF simple. Mais lorsque l'adaptation est poussée à son maximum, les bruits en vitesse atteint les niveaux de ce dernier. Il est d'ailleurs à noter que l'amplitude de l'adaptation est plus forte pour les axes horizontaux que verticaux mais qu'elle s'active plus facilement sur cette dernière. Enfin, comme précédemment, les termes non-diagonaux reliant position et vitesse dans la même direction sont les seuls à être significatifs, en revanche toutes les corrélations sont maintenant négatives.

NNAKF

Enfin, intéressons nous au NNAKF. Il est ici beaucoup plus compliqué d'étudier en détail le fonctionnement de l'adaptation puisqu'elle inclut un réseau de neurones et que les trois axes ne sont pas gérés indépendamment. On peut toutefois représenter comme précédemment la matrice de bruit de base ainsi que l'amplitude de l'adaptation correspondant à l'activation de toutes les matrices apprises, voir figure 3.8.

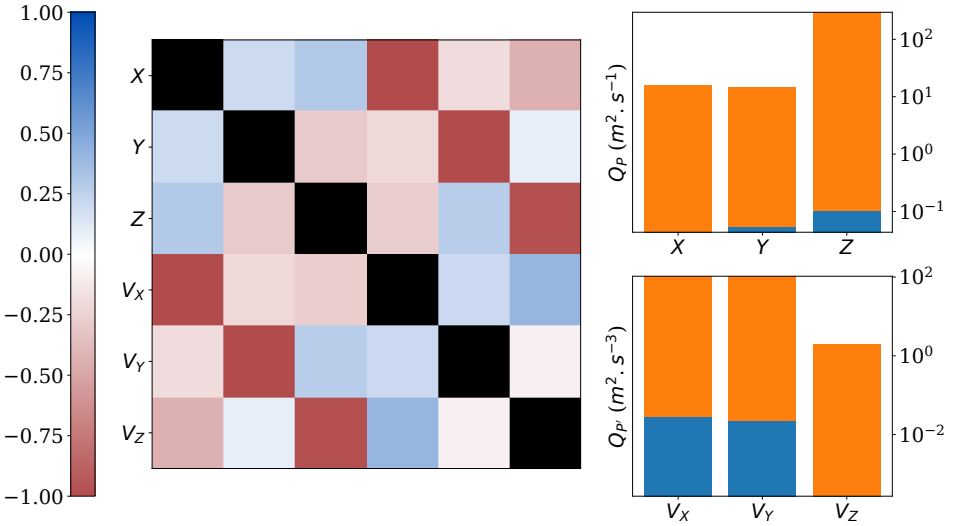


FIGURE 3.8 – Représentation de la matrice de bruit de modèle apprise pour le NNAKF. La matrice sur la gauche représente les coefficients de corrélation entre les différentes variables de position et de vitesse pour le niveau de bruit minimal. Les diagrammes à droite représentent en bleu les valeurs de variance de ces variables et en orange l'amplitude de l'adaptation sur une échelle logarithmique.

L'amplitude des variances est ici bien plus importante que pour l'adaptation de Castella, lui permettant d'augmenter le bruit de base d'un facteur 100, voire 1000 pour l'altitude, lorsqu'elle est poussée au maximum. Toutefois, le fonctionnement du réseau de neurones restant encore imparfaitement compris, rien ne nous permet encore d'affirmer que les valeurs maximales et minimales puissent être effectivement atteintes. L'étude des scénarios de robustesse de la section 3.4.4 nous permettra d'en savoir plus sur le fonctionnement réel de l'adaptation.

3.4.4 Robustesse

Robustesse au bruit

Dans cette expérience, nous étudions la robustesse du NNAKF lorsque l'on joue sur l'amplitude du bruit de mesure, et nous le comparons au KF simple et

à l'adaptation de Castella. Pour cela des trajectoires en ligne droite sont simulée partant toutes de l'origine du repère et partant dans les six directions des axes à différentes vitesses ($1m.s^{-1}$, $10m.s^{-1}$, $100m.s^{-1}$ $1000m.s^{-1}$). On génère ensuite les détections à fournir au filtre en bruitant les positions des trajectoires. Dans chaque scénario un axe est sélectionné pour être bruité tout au long de la trajectoire avec une amplitude allant de $10m$ à $10km$. Le bruit sur les autres axes est fixé à une amplitude de $10m$. On mesure la précision de chaque filtre sur chaque scénario avec la REQM en position sur l'axe bruité dans celui-ci.

En observant les valeurs de REQM en fonction de l'amplitude du bruit sur une double échelle logarithmique, l'ensemble des mesures semblent alignés autour d'une même droite comme le montre la figure 3.9, indiquant que la REQM suit une loi de puissance de type REQM = $e^{\beta}\sigma^{\alpha}$ où α et β sont les coefficients de la régression linéaire. On peut aussi remarquer que l'erreur est en général moins importante en utilisant le NNAKF que l'adaptation de Castella, elle même plus précise que le KF simple, mais que les écarts relatifs des erreurs tend à se réduire lorsque le bruit augmente. Dans les tableaux 3.6, 3.7 et 3.8, nous cherchons l'influence que peuvent avoir les trois autres paramètres des scénarios sur les résultats, à savoir la direction de la trajectoire, la vitesse de celle-ci, et l'axe bruité.

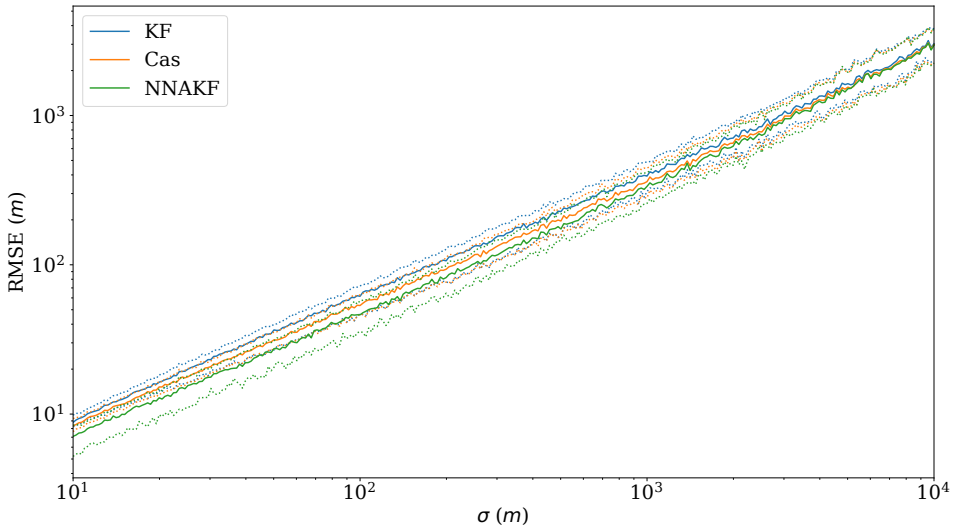


FIGURE 3.9 – Évolution de la REQM en position des différents filtres en fonction de l'intensité du bruit de mesure. Les courbes en trait plein représentent les médianes et celles en pointillés les premier et dernier déciles.

Dans les tableaux 3.6 et 3.7 on peut voir que les coefficients de la régression sont pratiquement identiques quelque soit la direction ou la vitesse de la trajectoire, mais varient selon le filtre utilisé. On en déduit que l'influence de ces facteurs sur les erreurs des filtres est négligeable. On peut cependant, grâce au tableau 3.7 constater une légère tendance à la baisse de e^{β} (et donc de l'erreur pour des bruits faibles) lorsque la vitesse augmente, mais cette évolution n'est

pas significative.

Contrairement aux deux autres facteurs précédents, le choix de l'axe sur lequel on augmente le bruit a une influence relativement nette sur les valeurs d'erreur, comme on peut le voir dans le tableau 3.8. Elle est moins importante lorsque c'est l'axe vertical qui est bruité que lorsque c'est l'un des axe horizontaux. L'écart relatif est de 25% pour le KF simple, 8% pour l'adaptation de Castella et de 35% pour le NNAKF. Dans tous les cas le NNAKF reste plus performant que les autres filtres.

Filtre	Direction	α	e^{β} (m)	R^2
KF	$\vec{e}_X$	0.832	1.30	0.988
	$-\vec{e}_X$	0.832	1.30	0.988
	$\vec{e}_Y$	0.831	1.30	0.989
	$-\vec{e}_Y$	0.832	1.30	0.989
	$\vec{e}_Z$	0.835	1.29	0.988
	$-\vec{e}_Z$	0.833	1.30	0.989
Castella	$\vec{e}_X$	0.837	1.16	0.990
	$-\vec{e}_X$	0.838	1.15	0.991
	$\vec{e}_Y$	0.838	1.16	0.991
	$-\vec{e}_Y$	0.838	1.16	0.991
	$\vec{e}_Z$	0.841	1.15	0.990
	$-\vec{e}_Z$	0.839	1.16	0.991
NNAKF	$\vec{e}_X$	0.869	0.86	0.987
	$-\vec{e}_X$	0.870	0.85	0.987
	$\vec{e}_Y$	0.868	0.86	0.987
	$-\vec{e}_Y$	0.870	0.86	0.987
	$\vec{e}_Z$	0.872	0.85	0.987
	$-\vec{e}_Z$	0.870	0.86	0.987

TABLEAU 3.6 – Coefficients de la régression linéaire de la REQM en fonction du niveau de bruit sur une double échelle logarithmique ainsi que le R^2 de celle-ci pour les différents filtres et selon les directions possibles de la trajectoire.

Filtre	Vitesse $(m.s^{-1})$	α	e^{β} (m)	R^2
KF	1	0.831	1.31	0.989
	10	0.832	1.30	0.988
	100	0.833	1.29	0.989
	1000	0.834	1.29	0.988
Castella	1	0.837	1.17	0.991
	10	0.838	1.16	0.990
	100	0.839	1.15	0.991
	1000	0.840	1.15	0.990
NNAKF	1	0.868	0.87	0.987
	10	0.869	0.86	0.987
	100	0.870	0.85	0.987
	1000	0.872	0.85	0.986

TABLEAU 3.7 – Coefficients et R^2 selon les vitesses possibles de la trajectoire.

Filtre	Axe bruité	α	e^{β} (m)	R^2
KF	R_{XX}	0.829	1.42	0.995
	R_{YY}	0.830	1.42	0.995
	R_{ZZ}	0.839	1.07	0.989
Castella	R_{XX}	0.839	1.19	0.993
	R_{YY}	0.840	1.19	0.993
	R_{ZZ}	0.838	1.09	0.989
NNAKF	R_{XX}	0.854	0.99	0.991
	R_{YY}	0.855	0.99	0.991
	R_{ZZ}	0.901	0.64	0.987

TABLEAU 3.8 – Coefficients et R^2 selon les directions possibles du bruitage.

Pour bien comprendre ce qui est observé dans cette expérience, il faut se rendre compte que la meilleure stratégie d'adaptation du bruit de modèle ici serait de le mettre à zéro. puisque la trajectoire respecte très exactement le modèle cinématique des filtres. Ce qui est mesuré ici est en fait la capacité des filtres à réduire leur bruit de modèle lorsque la situation l'exige malgré un fort bruitage. Pour l'adaptation de Castella comme pour le NNAKF, l'indicateur utilisée pour adapter le bruit est l'innovation normalisée qui tend vers 1 lorsque le bruit augmente. Pour cette valeur, l'adaptation de Castella ne prend que 4.9% de sa valeur maximale pour les axes horizontaux et 8.4% pour l'axe vertical, comme le montre la figure 3.7b, ce niveau de bruit additionnel est du même ordre de grandeur que celui avant adaptation. Pour le NNAKF, les coefficients qui activent les

matrices de bruit ne sont pas constants au cours de la trajectoire en raison de la mémoire du LSTM, mais ils convergent rapidement vers des valeurs qui ne dépassent pas les quelques centièmes non plus comme on peut le voir dans la figure 3.10. On peut conclure de cette dernière analyse que les réglages de l'adaptation de Castella et du NNAKF sont robustes aux bruits de mesure.

Robustesse pour une trajectoire uniformément accélérée

Dans cette expérience, nous étudions la robustesse du NNAKF lorsqu'il doit pister une trajectoire uniformément accélérée tout en gardant un modèle cinématique à accélération nulle, il sera encore comparé au KF simple et à l'adaptation de Castella. Les trajectoires considérées démarrent de l'origine du repère avec une vitesse nulle. Elles sont ensuite accélérées uniformément dans une des six directions du repère avec une valeur allant de $0.001g$ à $10g$ où g est l'accélération de la pesanteur à la surface de la Terre ($g = 9.80665 m.s^{-2}$). Toutes les détections sont obtenues avec un bruit gaussien sphérique d'amplitude $10m$ sur chaque axe. Pour chaque scénario, on mesure la précision des filtres avec la REQM en position ou en vitesse sur l'axe de l'accélération.

Sur la figure 3.11, on a représenté l'évolution de l'erreur en position lorsque l'accélération augmente dans les différentes directions pour les différents filtres. Lorsque l'accélération est faible (entre $10^{-3}g$ et $10^{-2}g$), la figure ne permet pas de distinguer clairement les filtres entre eux, l'erreur dépendant plus du caractère aléatoire du bruit de mesure que du type de filtre ou de la valeur de l'accélération, on peut toutefois noter que le NNAKF représenté en pointillés semble faire moins d'erreur que les autres filtres. À partir de $10^{-1}g$, l'erreur de prédiction qui vaut $\frac{1}{2}a\Delta t^2$ où a est l'accélération et Δt, atteint l'ordre de grandeur de l'erreur de mesure qui est de $10m$. Au delà l'erreur décolle lorsque le bruit de modèle n'arrive plus à compenser le retard de la prédiction sur la trajectoire et suit une loi de puissance dont le coefficient avoisine la valeur 1 après régression. Autrement dit, lorsque l'accélération est suffisamment forte, tous les filtres font une erreur proportionnelle à celle-ci. En revanche le facteur, et donc l'accélération limite à laquelle le filtre décroche, dépend à la fois de la direction et du filtre employé. Ainsi, on peut dire que les filtres sont plus robustes aux accélérations horizontales que verticales et que le NNAKF est plus robuste que l'adaptation de Castella, elle-même plus robuste que le KF simple. Toutefois la comparaison entre NNAKF et Castella est à nuancer car la figure 3.12 montre qu'en ce qui concerne la précision sur la vitesse verticale, c'est l'adaptation de Castella qui est la plus robuste.

On peut comme précédemment s'intéresser aux valeurs que prennent les innovations normalisées dans ces scénarios, ainsi qu'à leur impact sur l'adaptation. Lorsque l'accélération tend vers l'infini, la valeur de l'innovation sur l'axe accéléré tend elle aussi vers l'infini, mais reste stable sur les autres axes. Ainsi l'adaptation de Castella s'active complètement sur l'axe nécessaire, mais pas sur les autres, la robustesse du filtre pour ces scénarios sera donc d'autant plus forte

que ce bruit limite est haut. Concernant le NNAKF, les règles d'activation des matrices de bruit sont nettement plus compliquées étant donné qu'elles sont gérées par un réseau de neurones et que les axes ne sont pas traités indépendamment. Toutefois, des simulations numériques en utilisant des innovations normalisées prenant une valeur forte sur un axe et des valeurs faibles sur les autres font apparaître une convergence rapide des coefficients d'activation. Ces limites ne prennent que des valeurs identifiables parmi 0, 1, $\frac{1}{2}$, $\sigma(1)$ et $\mathrm{W}(1)$ où σ est la fonction sigmoïde et $\mathrm{W}(1)$ est l'unique solution réelle de $z = \mathrm{W}(z)e^{\mathrm{W}(z)}$ pour $z = 1$. On peut en déduire une matrice de bruit d'adaptation limite pour des trajectoires uniformément accélérées pour chacun des trois axes, c'est ce qui est fait dans la figure 3.13. Lorsque l'accélération est mise sur l'un des axes horizontaux, ce sont les deux axes qui sont bruités. Si ce choix peut paraître inapproprié pour ce type de scénario, il ne l'est pas forcément pour les trajectoires de la base de données. En effet, les erreurs de modélisations peuvent aussi y être dues à un virage, au quel cas les deux axes sont impactés. Pour une accélération verticale, le NNAKF active bien l'adaptation sur l'axe correspondant, mais aussi sur les vitesses horizontales. Ce défaut rejoint les mauvaises performances de ce NNAKF pour ce type de scénario évoqué à la fin du paragraphe précédent, et peut s'expliquer par le fait que l'évolution de l'altitude dans les trajectoires de la base de données est gérée par des pics de vitesse verticale et ne présente donc pas d'accélération qui dure dans le temps.

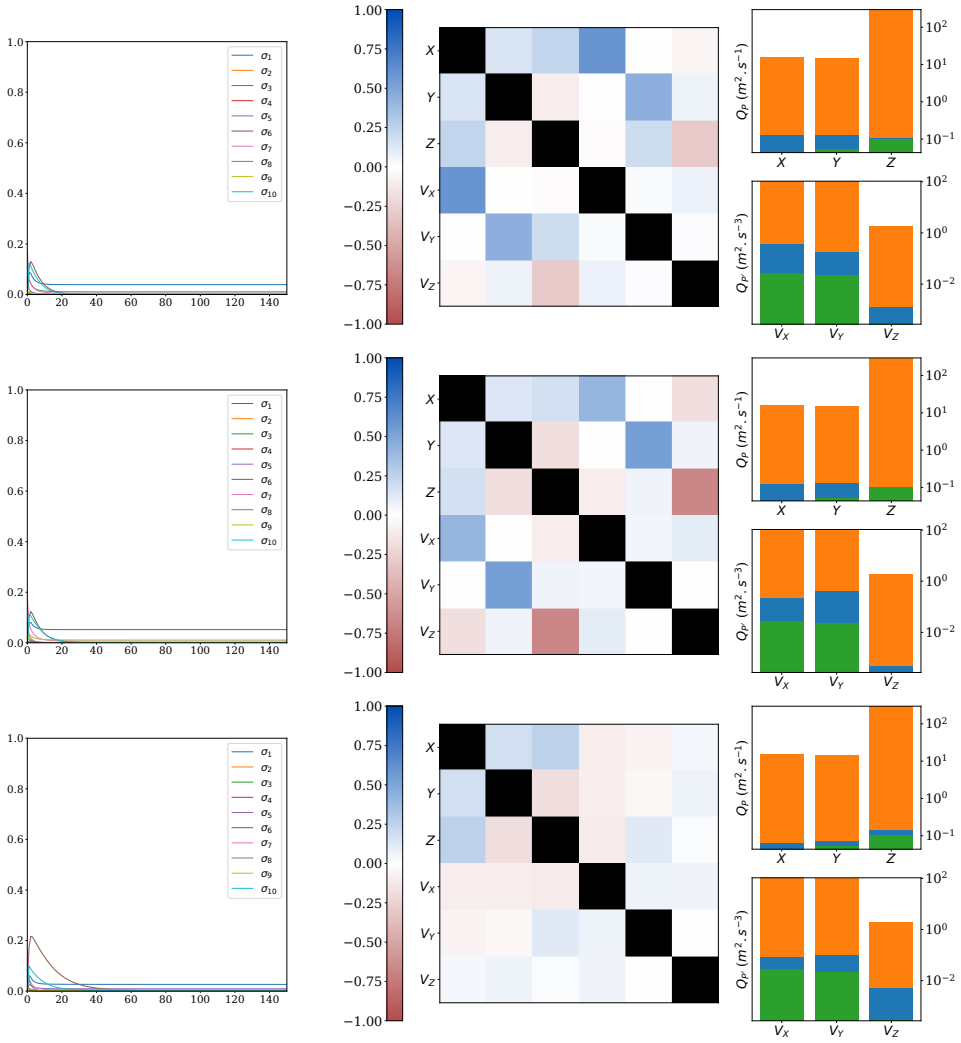


FIGURE 3.10 – Chaque ligne représente l'évolution de l'adaptation du NNAKF à un scénario CS fortement bruité sur l'un des axes, respectivement X, Y et Z. Les courbes de gauche représentent l'évolution des coefficients exprimés par le LSTM au cours des cycles du filtre et montrent une convergence vers des niveaux faibles. Les figures de droite montrent les matrices de bruit de modèle limites pour les différents scénarios. Les barres en vert et orange représentent les valeurs extrêmes théoriques déjà définies dans la figure 3.8. Les barres bleues représentent les niveaux de variance réellement atteints par l'adaptation.

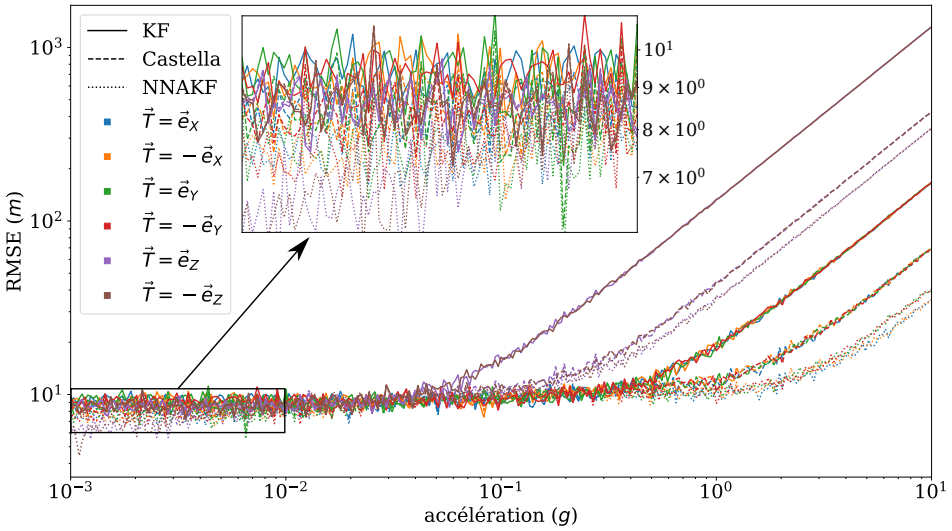


FIGURE 3.11 – Évolution de la REQM en position des différents filtres en fonction de l'accélération donnée aux trajectoires dans la direction $\vec{T}$.

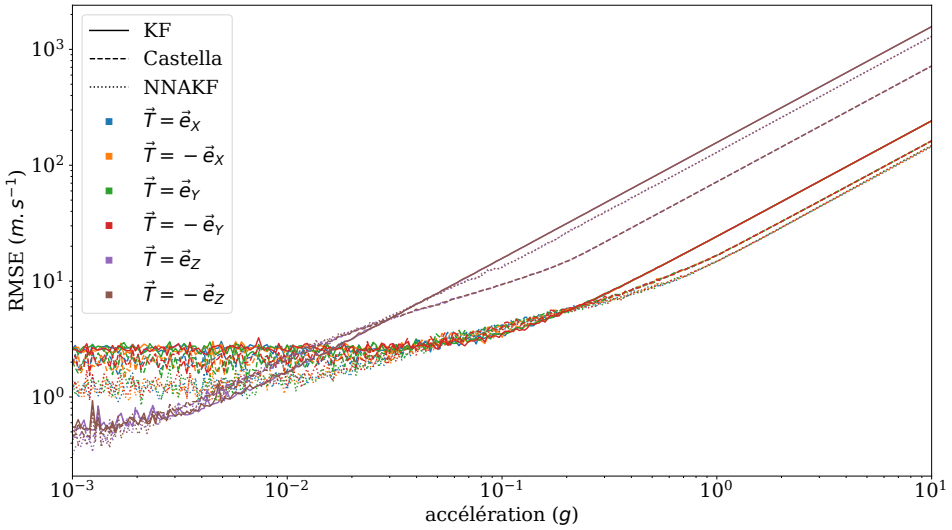


FIGURE 3.12 – Évolution de la REQM en vitesse des différents filtres en fonction de l'accélération donnée aux trajectoires dans la direction $\vec{T}$.

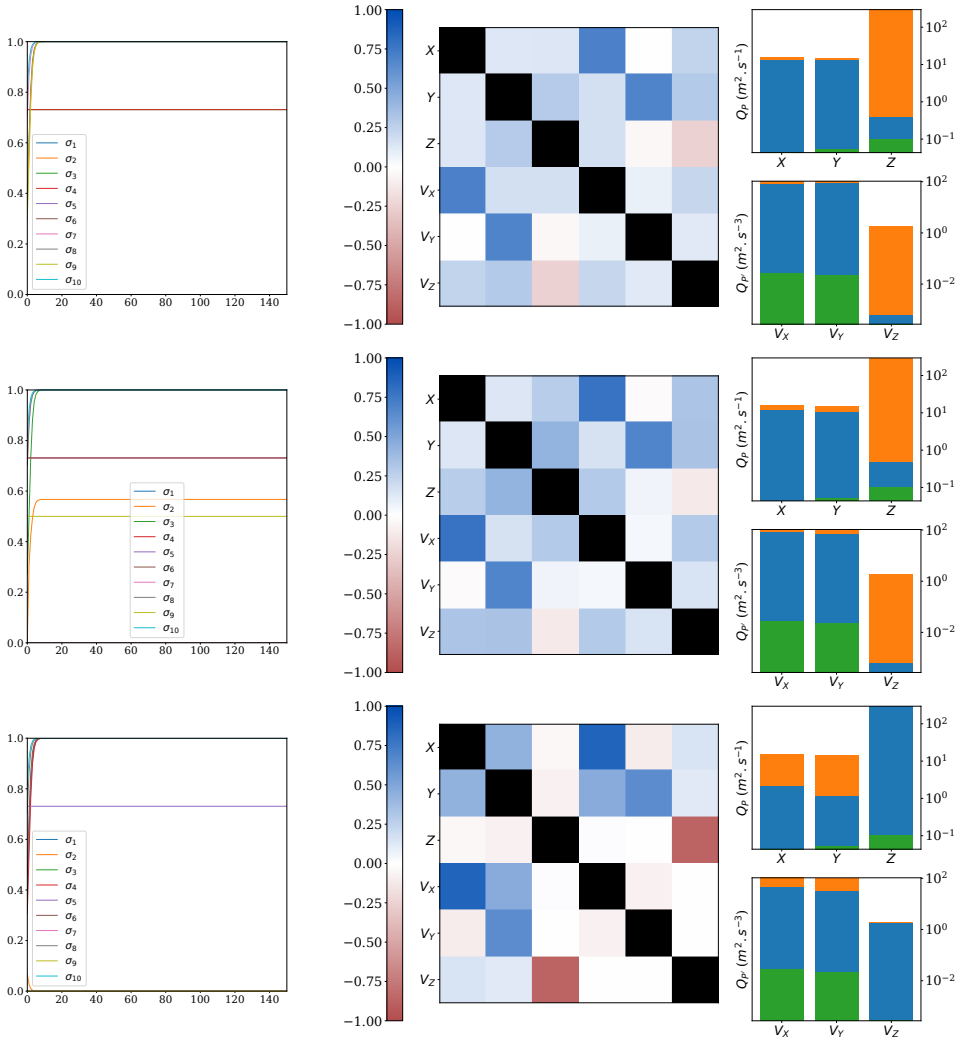


FIGURE 3.13 – Chaque ligne représente l'évolution de l'adaptation du NNAKF à un scénario MUA fortement accéléré sur l'un des axes, respectivement X, Y et Z. Les courbes de gauche représentent l'évolution des coefficients exprimés par le LSTM au cours des cycles du filtre et montrent une convergence vers des niveaux spécifiques. Les figures de droite montrent les matrices de bruit de modèle limites pour les différents scénarios. Les barres en vert et orange représentent les valeurs extrêmes théoriques déjà définies dans la figure 3.8. Les barres bleues représentent les niveaux de variance réellement atteints par l'adaptation.

3.4.5 Apprentissage

Tous les filtres présentés dans ce chapitre sont implémentés sous la forme de réseaux récurrents et l'entraînement de tels algorithmes peut parfois s'avérer délicat PASCANU et collab. [2013]. C'est pourquoi dans cette section nous nous intéresserons plus en détail au déroulement de la phase d'apprentissage. La figure 3.14 représente l'évolution de la fonction de coût au cours de l'entraînement des différentes variations du NNAKF ainsi que du KF simple et de l'adaptation de Castella.

On peut dans un premier temps constater que les filtres classiques (KF et Castella) présentent des apprentissages relativement peu sensibles aux conditions initiales, toutes les tentatives suivent à peu près la même progression régulière et convergent à la même vitesse vers la même limite. Il n'est pas non plus inintéressant de noter que l'entraînement de l'adaptation de Castella semble se faire en deux temps avec la présence d'un palier intermédiaire au cours duquel la progression est ralentie et pour lequel la fonction de coût avoisine celle de la limite du KF simple.

Concernant le NNAKF utilisant une simple couche de neurones (FC), ce même palier semble présent mais bien moins marqué que pour l'adaptation de Castella. L'apprentissage peut aussi sembler reproductible à première vue, mais un zoom sur la figure 3.15 permet de constater que ce n'est pas le cas lorsque le nombre de matrices d'adaptation est fixé à six. En effet il semble que deux valeurs de convergence distinctes sont possibles pour la fonction de coût, l'une avoisinant celle obtenue pour le filtre à trois et l'autre celle de celui à dix matrices. Ce phénomène peut être dû à la présence d'un minimum local dans la fonction de coût, il est donc important d'effectuer plusieurs tentatives d'entraînement avec des conditions initiales différentes afin d'augmenter nos chances d'approcher le minimum global.

Tout en gardant un modèle cinématique CS, ce problème ne semble pas se manifester lorsque le NNAKF utilise un LSTM pour gérer son adaptation. Il réapparaît toutefois pour les modèles MUA et Singer lorsque le nombre de matrices d'adaptation est limité à trois. On remarque aussi que la progression pour le modèle MUA est loin d'être régulière, il semble que les performances de ces filtres soient très sensibles aux variations des paramètres entraînables.

Un autre facteur à prendre en compte pour l'apprentissage est le temps de calcul nécessaire. Le tableau 3.9 récapitule ces temps pour les différents filtres présentés dans ce chapitre. On peut y constater que l'ajout d'une adaptation augmente le temps d'apprentissage mais que l'on reste dans le même ordre de grandeur, contrairement à l'utilisation d'un modèle complexe comme celui de Singer pour lequel l'apprentissage s'avère bien plus coûteux.

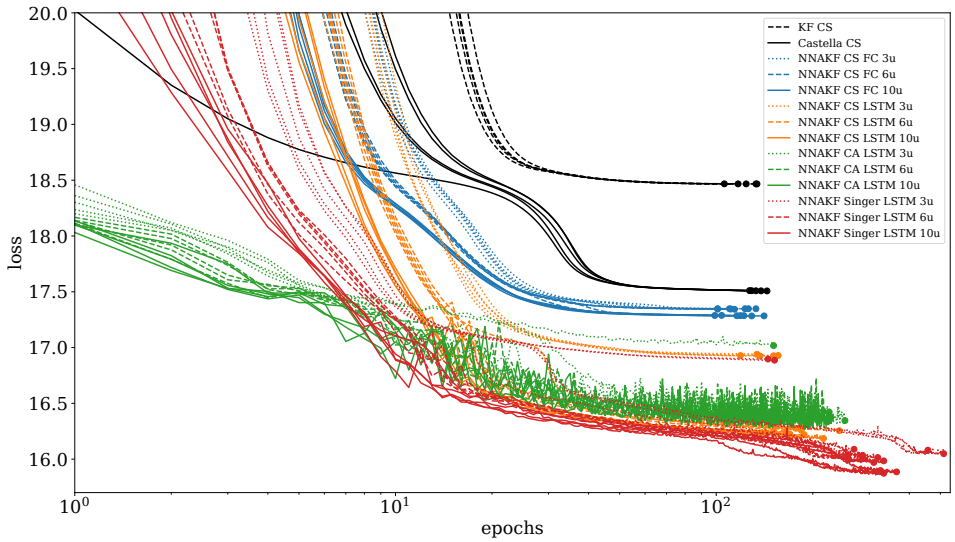


FIGURE 3.14 – Évolution de la fonction de coût au cours de l'apprentissage des différents filtres étudiés. L'abscisse correspond au nombre d'*epochs* écoulées sur une échelle logarithmique. Pour chaque type de filtre, cinq tentatives d'apprentissage ont été effectuées avec des conditions initiales tirées aléatoirement. Le point à la fin de chaque courbe permet de distinguer les temps des déclenchements de la condition d'arrêt lorsque celles-ci convergent ensemble.

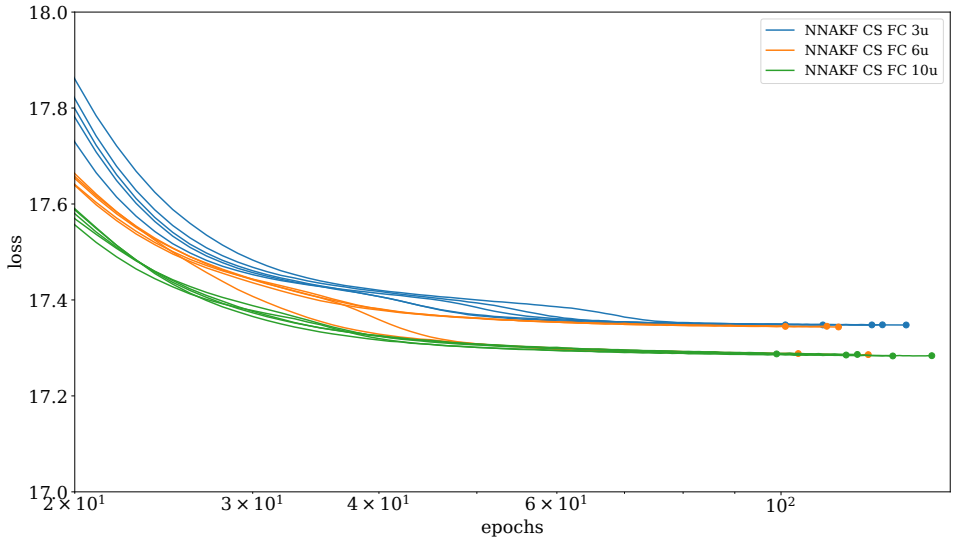


FIGURE 3.15 – Évolution de la fonction de coût au cours de l'apprentissage des NNAKF utilisant un modèle CS, une simple couche de neurones pour l'adaptation et trois, six ou dix matrices d'adaptation. Il s'agit des courbes déjà présentes en bleu sur la figure 3.14.

Filtre	# de paramètres	# d'*epochs*	Temps par *epoch* (s)	Temps d'apprentissage
KF CS	36	124	53.96	$1h\,52'$
Castella CS	45	133	119.45	$4h\,25'$
NNAKF CS FC 3u	156	123	104.75	$3h\,35'$
NNAKF CS FC 6u	276	111	109.19	$3h\,22'$
NNAKF CS FC 10u	436	119	109.85	$3h\,38'$
NNAKF CS LSTM 3u	234	137	119.60	$4h\,33'$
NNAKF CS LSTM 6u	504	184	117.08	$5h\,59'$
NNAKF CS LSTM 10u	976	170	117.54	$5h\,33'$
NNAKF MUA LSTM 3u	417	187	189.17	$9h\,50'$
NNAKF MUA LSTM 6u	822	212	189.66	$11h\,10'$
NNAKF MUA LSTM 10u	1474	159	179.41	$7h\,55'$
NNAKF Singer LSTM 3u	418	269	368.71	$27h\,33'$
NNAKF Singer LSTM 6u	823	286	436.64	$34h\,41'$
NNAKF Singer LSTM 10u	1475	333	439.72	$40h\,40'$

TABLEAU 3.9 – Nombre de paramètres entraînables, nombre médian d'*epochs* par apprentissage, temps médian par *epoch* et temps total correspondant. Ces valeurs ont été obtenues après cinq tentatives d'apprentissage pour chacun des filtres présentés. Les temps dépendent bien entendu de la machine utilisée et sont donc donnés à titre de comparaison.

3.5 Conclusion

Dans ce chapitre, nous avons proposé un filtre de Kalman adaptatif faisant intervenir des réseaux de neurones appelé NNAKF. Ses performances une fois réglé surpassent celles du KF classique ainsi que celles de l'adaptation de Castella pour un coût computationnel relativement faible. De plus, contrairement à cette dernière, ses capacités peuvent être étendues par une augmentation du nombre de matrices de bruits pouvant servir à l'adaptation. Cependant, si le NNAKF s'avère supérieur dans la catégorie des filtres adaptatifs n'utilisant qu'un seul modèle cinématique, les IMM bien que généralement moins économes sont plus souvent utilisés dans les algorithmes de pistage et sont généralement considérés comme l'état de l'art en matière de filtrage. Ils feront l'objet du chapitre suivant.

3.6 Références

CASTELLA, F. R. 1980, «An adaptive two-dimensional kalman tracking filter», *IEEE Transactions on Aerospace and Electronic Systems*, , nº 6, p. 822–829. 41, 43

COSKUN, H., F. ACHILLES, R. DIPIETRO, N. NAVAB et F. TOMBARI. 2017, «Long short-term memory kalman filters : Recurrent neural estimators for pose regularization», dans *Proceedings of the IEEE International Conference on Computer Vision*, p. 5524–5532. 50

JOUABER, S., S. BONNABEL, S. VELASCO-FORERO et M. PILTÉ. 2021, «Nnakf : A neural network adapted kalman filter for target tracking», dans *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, p. 4075–4079. 43

PASCANU, R., T. MIKOLOV et Y. BENGIO. 2013, «On the difficulty of training recurrent neural networks», dans *International conference on machine learning*, p. 1310–1318. 62

SINGER, R. A. 1970, «Estimating optimal tracking filter performance for manned maneuvering targets», *IEEE Transactions on Aerospace and Electronic Systems*, , nº 4, p. 473–483. 46

VAIDEHI, V., N. CHITRA, M. CHOKKALINGAM et C. KRISHNAN. 2001, «Neural network aided kalman filtering for multitarget tracking applications», *Computers & Electrical Engineering*, vol. 27, nº 2, p. 217–228. 49

ZEILER, M. D. 2012, «Adadelta : an adaptive learning rate method», *arXiv preprint arXiv :1212.5701*. 45, 49

Chapitre 4

Modèles multiples

Sommaire

4.1 Résumé

Ce chapitre propose une approche du filtrage par modèles multiples aidé par des méthodes d'apprentissage automatique soit parce que les algorithmes intègrent des réseaux de neurones, soit parce que les algorithmes eux-mêmes ont un trop grand nombre de paramètres pour qu'il soit envisageable de les régler à la main. Nous introduirons dans ce chapitre un nouveau modèle appelé modèle circulaire uniforme qui conserve une vitesse et un taux de virage constants, il s'agit évidemment d'un modèle qui ne peut être utilisé seul puisque tournant toujours dans le même sens. Par la suite une étude approfondie des filtres et de leur fonctionnement permettront de comprendre le rôle des différents modèles qui les composent et comment ils interagissent entre eux. Une extension facultative du réseau de neurones visant à résoudre le problème de la classification des cibles sera ensuite proposée. Enfin, les limitations de ce type de filtres, et notamment celle de sa robustesse seront mises en lumière.

4.2 Introduction

Dans le chapitre précédent, nous avons travaillé à partir de filtres de Kalman classiques sans sortir de leur cadre optimal en se restreignant à un modèle d'évolution linéaire. Cependant il serait illusoire de penser que toutes les cinématiques rencontrées pourront être suffisamment bien décrites par un modèle linéaire. Pour pallier cette limitation il existe plusieurs solutions. La première consiste à utiliser un modèle non-linéaire, une approximation des opérateurs du filtre est alors obtenue par développement au premier ordre des équations différentielles qui régissent la cinématique, il s'agit alors d'un EKF Gelb [1974]. Il est même possible d'aller jusqu'à prendre comme espace d'états non pas un espace vectoriel mais un groupe de Lie, on parle alors de filtre IEKF Barrau et Bonnabel [2016]. Une version de ces derniers basée sur un modèle à paramètres de Frenet constants a d'ailleurs été étudiée dans Pilté et collab. [2017]. Cependant la non-linéarité des modèles d'évolution de ces filtres, en plus de celle de la relation qui relie les mesures polaires des radars à l'espace cartésien des états, induit des instabilités qui peuvent faire diverger l'algorithme de filtrage ce peut vite compliquer voire rendre impossible le processus d'apprentissage automatique. Une autre solution, qui est celle que nous aborderons tout au long de ce chapitre est celle des modèles multiples comme proposée par Blom et Bar-Shalom [1988]. Cette solution, qui consiste à mettre en émulation plusieurs modèles aux cinématiques et/ou réglages différents, est d'ailleurs largement utilisée dans le domaine du pistage radar.

4.3 Interacting Multiple Models

4.3.1 Fonctionnement

Le principe de l'IMM détaillé dans la section 1.2.4 peut se résumer ainsi. Plusieurs Filtres de Kalman (FK) de modèles ou de paramètres différents sont utilisés parallèlement pour filtrer un même signal. Dans le même temps, la confiance en chacun de ces filtres est représentée par un vecteur de probabilités qui évolue au cours de la prédiction selon une matrice de transition prédéfinie. Les états initiaux des filtres sont eux mixés au début de l'étape de prédiction afin de les faire se rapprocher de ceux des filtres les plus probables. Enfin, les probabilités sont mises à jour pour favoriser les filtres ayant les prédictions les plus vraisemblables. L'étape d'estimation a lieu indépendamment dans chaque filtre de façon normale.

Pour notre problème en particulier, les paramètres que nous pourront régler automatiquement sont la matrice de transition, ainsi que les bruits et paramètres des modèles des différents filtres (α pour le modèle de Singer par exemple). Le nombre de filtres et le choix des modèles sont donc des hyperparamètres à fixer avant l'apprentissage. La matrice de transition sera le résultat de la régularisation sur chaque ligne par une fonction *softmax* d'une matrice de poids de dimension (N × N) où N est le nombre total de filtres.

Un premier exemple simple d'IMM à étudier serait uniquement composé de filtres ayant pour modèle cinématique le Mouvement Rectiligne Uniforme (MRU), il pourra comme ça être comparé aux filtres du chapitre précédent qui utilisent le même modèle. Les résultats d'un tel filtre à trois modèles MRU sont présentés dans le tableau 4.1. On peut y constater que cet IMM a des résultats assez proches des NNAKFs à 3 et 10 matrices de bruit. Tous ces filtres ont suivi le même entraînement décrit au chapitre précédent.

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
Oracle	28.26	13.66	2.50	3.37	NA
PbPO	30.66	31.13	1.63	6.40	8.134
FK	59.62	46.69	4.77	17.51	18.467
Castella	54.11	46.62	3.81	14.29	17.507
NNAKF MRU LSTM 3M	51.14	46.65	3.48	12.46	16.924
NNAKF MRU LSTM 10M	51.05	**42.82**	3.49	12.25	**16.182**
IMM 3MRU	**50.96**	46.38	**3.14**	**12.09**	16.772

TABLEAU 4.1 – REQM sur différentes métriques des filtres présentés au chapitre précédent ainsi que de l'IMM à trois modèles MRU. La fonction de coût obtenue pour chaque filtre est aussi représentée dans la dernière colonne.

4.3.2 Modèles cinématiques

Dans la suite de ce chapitre, nous allons enrichir l'IMM avec d'autres types de modèles. Le modèle de Singer et le Mouvement Uniformément Accéléré

(MUA) ainsi que deux autres qui ne peuvent être utilisés que dans un IMM seront aussi envisagés. Le premier est le modèle Arrêt qui suppose que la cible est immobile, ce qui se traduit par l'opérateur évolution $F(\Delta t) = I_3$ et le bruit de modèle $Q_{\Delta t} = \Delta t Q$. Le second est le Mouvement Circulaire Uniforme (MCU) qui suppose que la cible suit un taux de virage (à gauche) γ constant qui est un paramètre entraînable du modèle, au même titre que le coefficient α dans le modèle de Singer. Sa définition formelle se trouve en annexe A.3.4 et vérifie :

$$\frac{d}{dt}V_X = -\gamma V_Y \tag{4.1}$$

$$\frac{d}{dt}V_Y = \gamma V_X \tag{4.2}$$

4.4 Optimisation bayésienne

Ces cinq modèles ajoutent autant d'hyper-paramètres à l'IMM puisqu'il nous faut définir le nombre de filtres utilisant chacun d'entre eux à y intégrer. Ces valeurs ne peuvent pas être réglées automatiquement par descente de gradient et le nombre de combinaisons devient rapidement trop grand pour envisager de toutes les explorer. C'est pourquoi nous proposons une autre méthode d'optimisation pour le réglage des hyper-paramètres, l'optimisation bayésienne.

4.4.1 Principe

L'optimisation bayésienne repose sur l'inférence de la fonction de coût attendue pour chacune des combinaisons envisagées par un processus gaussien. Après une initialisation basée sur un petit nombre d'observations correspondant à quelques combinaisons sélectionnées aléatoirement, on sélectionne une combinaison à tester en fonction de l'inférence bayésienne de la fonction de coût en faisant un compromis entre exploration et exploitation. En l'occurrence, la combinaison sélectionnée sera celle qui maximise les chances d'obtenir un coût plus faible que le plus faible obtenu jusqu'ici. L'IMM correspondant à cette combinaison de modèles est alors entraîné en optimisant la matrice de transitions, ses matrices de bruits et les éventuels paramètres de ses modèles. Le résultat obtenu est ensuite ajouté aux observations précédentes. Une fois l'inférence bayésienne mise à jour avec cette nouvelle donnée, on recommence l'opération en boucle jusqu'à atteindre les conditions d'arrêt fixées.

4.4.2 Expérience numérique

L'inférence bayésienne requiert le choix d'un noyau. Après quelques essais sur un problème simplifié, il s'est avéré que la covariance de Matérn ($C_\nu(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}(\sqrt{2\nu}\frac{d}{\rho})^\nu K_\nu(\sqrt{2\nu}\frac{d}{\rho})$) avec un paramètre $\nu = \frac{5}{2}$ permettait une

bonne convergence. C'est donc ce noyau qui a été sélectionné pour la suite. Il nous reste alors à délimiter l'espace de recherche. Pour cela nous nous sommes limité à dix filtres pour chaque modèle, par analogie avec le nombre de matrices de bruit du NNAKF, sauf pour le modèle MCU dont la limite sera doublée pour permettre les virages à gauche et à droite pour chaque valeur de taux de virage γ et le modèle Arrêt qui lui sera limité à trois filtres.

Pour notre expérience, les 20 premières combinaisons d'hyper-paramètres ont été sélectionnées aléatoirement avant de passer le relais à l'optimisation bayésienne. L'entraînement de chaque IMM a duré entre 6 heures et 3 jours selon la combinaison testée. Après quelques mois, ce sont 86 combinaisons sur les 111803 possibles qui ont été testées. Les dix meilleures figurent sur le tableau 4.2.

On peut dans un premier temps constater que ces dix résultats sont assez proches, il est donc possible que continuer à tester de nouvelles combinaisons n'apporte pas grand chose. De plus ces résultats surpassent aisément ceux du NNAKF et de tous les filtres à modèle unique que nous avons testés jusqu'ici. Dans un second temps, on peut aussi remarquer que tous ces IMM ont en commun d'utiliser au moins 10 filtres à modèle MCU, aucun des autre modèles n'est systématiquement utilisé dans ce palmarès, et le meilleur IMM utilise même exclusivement des modèles MCU. Par la suite, nous n'utiliserons donc plus que des IMM à 20 modèles MCU.

Arrêt	MRU	MCU	Singer	MUA	Fonction de coût
0	0	20	0	0	14,5253
0	0	12	6	0	14,5271
3	4	10	0	5	14,5346
3	8	13	3	4	14,5408
0	10	20	0	10	14,5600
2	8	13	2	6	14,5765
0	10	20	10	0	14,5893
0	10	20	0	5	14,6036
2	8	14	3	5	14,6059
3	0	20	10	10	14,6476

TABLEAU 4.2 – Tableau rassemblant les dix meilleurs IMM obtenus en terme de fonction de coût à l'issue de l'optimisation bayésienne.

4.4.3 Conclusions sur l'IMM et l'optimisation bayésienne

Dans cette section, nous avons exploré l'utilisation de méthodes d'apprentissage automatique pour des filtres IMM que l'on peut considérer comme classique dans la mesure où l'algorithme est utilisé tel qu'il a été proposé originellement. En revanche, le grand nombre de modèles employés par les filtres étudiés ici les rend pratiquement inutilisables sans méthode d'apprentissage automatique pour les régler.

Le principal inconvénient de l'IMM par rapport au NNAKF étudié dans la partie précédente est le temps de calcul nécessaire à son utilisation qui peut s'avérer limitant pour une utilisation embarquée sur un appareil léger. Le temps nécessaire à l'apprentissage peut lui aussi être plus long selon le nombre de modèles, mais ce qui rallonge le plus les temps de calcul est le choix des hyperparamètres par optimisation bayesienne qui nécessite une approche de type essai et erreur. Cette dernière nous a tout de même permis de mettre en évidence l'importance de modèles à virage afin d'améliorer la précision du filtrage, et même que leur utilisation seule est suffisante à atteindre les meilleures performances parmi les différentes combinaisons sur le jeu de données utilisé.

4.5 Étude du meilleur IMM obtenu

L'IMM à 20 filtres MCU obtient une fonction de coût (*loss*) plus faible que celle de chacun des autres filtres étudiés précédemment, et ce résultat s'observe aussi sur les mesures de REQM dans le tableau 4.3. Bien que ce filtre n'utilise pas de réseau de neurones, il serait inenvisageable de le régler manuellement au vu du grand nombre (1140) de paramètres qui le compose. Les méthodes d'apprentissage automatiques sont donc nécessaire pour l'utilisation de ce filtre. Dans la suite de cette partie, nous nous intéresserons au fonctionnement de celui-ci.

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
Oracle	28.26	13.66	2.50	3.37	NA
PbPO	30.66	31.13	1.63	6.40	8.134
FK	59.62	46.69	4.77	17.51	18.467
Castella	54.11	46.62	3.81	14.29	17.507
NNAKF MRU LSTM 10M	51.05	42.82	3.49	12.25	16.182
IMM 20MCU	**44.46**	**40.35**	**2.55**	**10.65**	**14.525**

TABLEAU 4.3 – REQM sur différentes métriques des filtres présentés au chapitre précédent ainsi que de l'IMM à trois modèles MRU. La fonction de coût obtenue pour chaque filtre est aussi représentée dans la dernière colonne.

4.5.1 Étude des taux de virage

La première observation à faire pour étudier le fonctionnement de cet IMM à 20 modèles concerne les taux de virages qui ont été réglés au cours de la phase d'apprentissage. Ces valeurs sont représentées sur la figure 4.1. On remarque que cinq des filtres ont un taux de virage négatif important, quatre en ont un positif, et les onze autres en ont un proche de zéro. Parmi ces derniers, le taux le plus important est de l'ordre de 10^{-5} $rad.s^{-1}$ ce qui correspond environ à un tour complet par semaine, on peut donc considérer qu'il s'agit de modèles MRU. Ainsi, la phase d'apprentissage permet effectivement d'obtenir des modèles MRU comme cas limite des modèles MCU, ce qui nous conforte dans l'idée de n'utiliser que ce type de modèle. Concernant les filtres présentant réellement

une rotation, le taux le plus fort pour chacun des sens est d'environ 0.08 $rad.s^{-1}$ et correspond à un tour complet en un peu plus d'une minute, puis décroît exponentiellement, le rapport entre deux taux de virage consécutifs est d'environ deux.

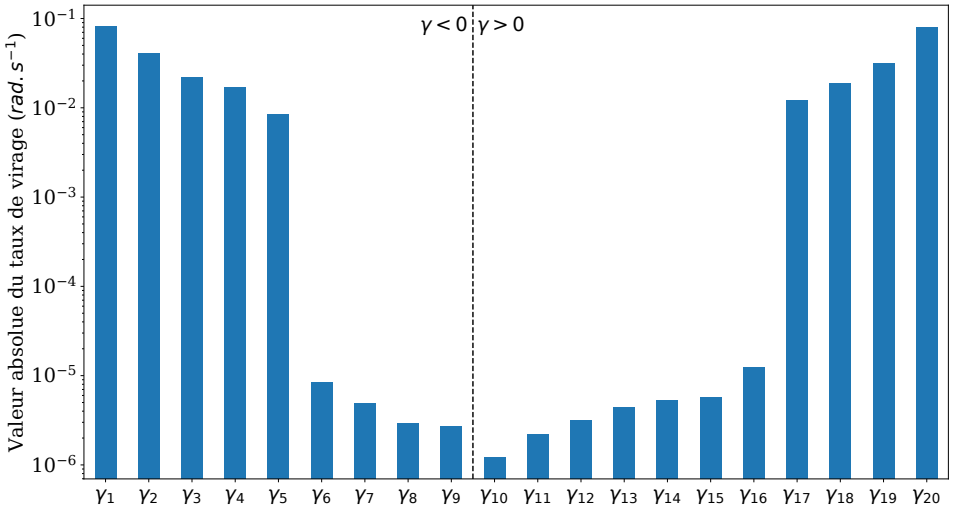


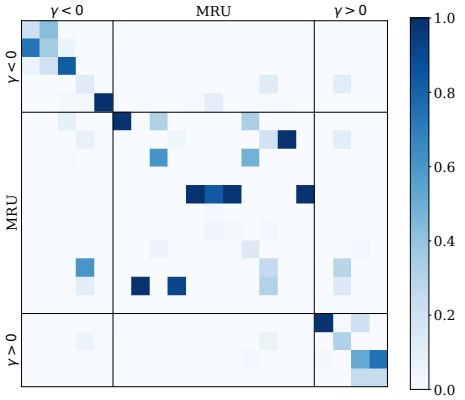
FIGURE 4.1 – Taux de virage des 20 modèles MCU de l'IMM classés dans l'ordre croissant de leur taux de virage sur une échelle logarithmique. La barre verticale sépare les valeurs positives et négatives.

4.5.2 Étude de la matrice de transition

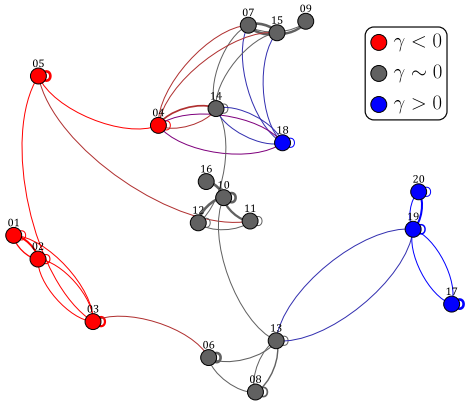
La matrice de transition apprise par l'IMM, représentée dans la figure 4.2a, est creuse. Il est donc possible de représenter les transitions sous forme d'un graphe. Dans le graphe de la figure 4.2b, seules les transitions avec une probabilité de plus de 1% sont représentées sous formes de liens entre les nœuds qui eux représentent chacun un filtre. La couleur du nœud dépend du taux de virage du filtre associé.

Tout d'abord, même si elle n'est pas parfaite, on peut observer une certaine symétrie entre les MCUs présentant des virages à gauche et à droite. On a en effet trois filtres dans chaque sens (01/02/03 et 20/19/17) qui présentent des transitions entre eux, surtout lorsque leur taux de virage sont proches, et on retrouve aussi les filtres 04 et 18 qui font office de liaison avec le filtre MRU pivot 14.

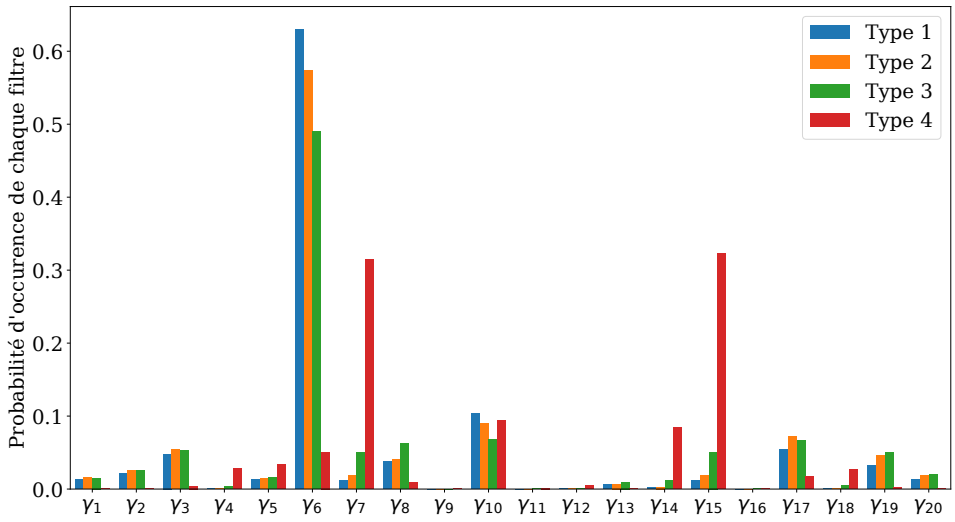
Concernant les MRUs justement, on peut distinguer trois clusters : 07/09/15, 10/11/12/16 et 06/08/13. Pour les deux derniers, on trouve un filtre stable au sein de chaque cluster vers lequel les probabilités des autres vont être transférées, à savoir le 10 et le 06. Pour le premier, le filtre 09 renvoie au filtre 15 qui forme avec le filtre 07 un cycle stable, les probabilités vont alterner entre ceux-ci. On peut donc considérer que chacun de ces clusters prend la forme d'un état ou cycle stable autour duquel gravitent un ou plusieurs états transitoires.



(a) Matrice de transition de l'IMM à 20 filtres MRU



(b) Représentations des transitions de l'IMM à 20 filtres MRU sous forme de graphe orienté



(c) Fréquence d'utilisation des différents filtres de l'IMM pour chacun des 4 types de trajectoires

FIGURE 4.2

En parallèle, nous pouvons aussi regarder à quelle fréquence chacun des filtres est utilisé, et ce en différenciant le type de la trajectoire, comme dans la figure 4.2c. On peut y constater une très nette différence entre le type 4 et les trois autres dans l'emploi qui est fait des filtres de cet IMM. En fait on peut dire que la partie du haut du graphe de la figure 4.2b est dédiée au type 4 et que la partie du bas est dédiée aux types 1, 2 et 3, le cluster du filtre 10 étant lui commun à tous les types.

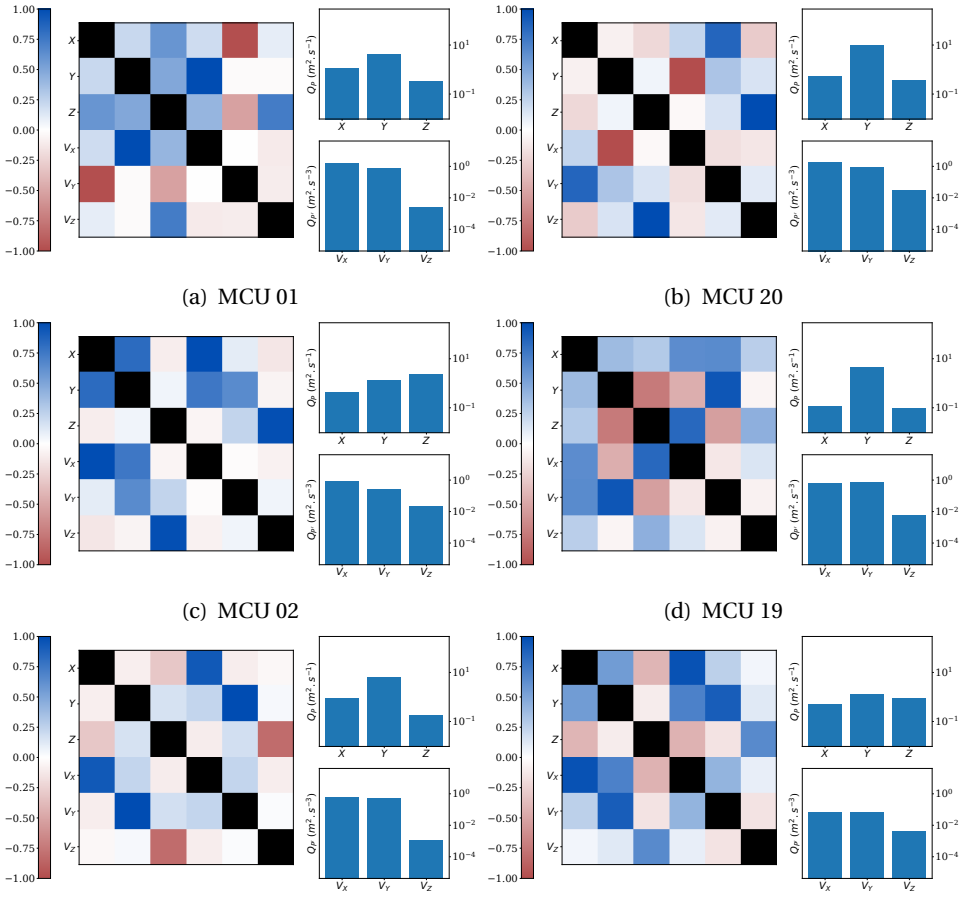
Une autre observation importante est l'absence presque totale des filtres 09, 11, 12, 13 et 16 dans l'utilisation qui est faite de l'IMM quelque soit le type de trajectoire, ce qui vient appuyer leur qualification d'état transitoire.

4.5.3 Étude des matrices de bruit

Après avoir vu comment s'articulaient les différents filtres de l'IMM, nous pouvons nous intéresser aux matrices de bruit apprises pour chacun d'entre eux. Tout d'abord, regardons les matrices correspondant aux filtres MCU dédiés aux types de trajectoires 1, 2 et 3. Ces matrices sont représentées dans la figure 4.3. Pour les virages les plus marqués (4.3a et 4.3b), on trouve de fortes corrélations entre X et V_Y, Y et V_X, et Z et V_Z. Si la corrélation entre position et vitesse sur l'axe vertical est toujours positive, celles entre les positions et vitesses des axes horizontaux orthogonaux sont de signes opposés, et dépendent du sens du virage. Cette forme particulière que prend le bruit de modèle vient appuyer le virage. Pour les virages les moins marqués, on retrouve une distribution avec de fortes corrélations entre position et vitesse dans la même direction, comme pour les modèles MRU du chapitre précédent. Toutefois cette observation est un peu moins claire pour les virages à gauche (4.3f) que pour les virages à droite (4.3e), ceci peut s'expliquer par le fait que ce filtre est utilisé à une fréquence non-négligeable pour des trajectoires de type 4, contrairement à son homologue de sens inverse. Enfin concernant les virages intermédiaires (4.3c et 4.3d), la forme de la distribution ne présente pas de caractéristique clairement identifiable et interprétable.

Ensuite, nous pouvons nous intéresser aux filtres à virage dédiés aux trajectoires de type 4 (4.4a et 4.4b) ainsi qu'à celui qui semble faire le lien entre ceux-ci (4.4c). Si les deux matrices des filtres à virage ne sont que très peu lisibles, celle du filtre 14 présente les mêmes caractéristiques que la matrice employée par le FK à simple modèle MRU.

Enfin, il nous reste à étudier les filtres MRU 07, 15, 10, 06 et 08 représentés dans la figure 4.5. Non seulement les filtres MRU 07 et 15 renvoient l'un vers l'autre dans les transitions avec une probabilité proche de 1, mais ils ont aussi des matrices de bruit presque identiques, fortement concentrées sur l'axe vertical. À l'inverse, les modèles MRU 06 et 08 présentent plus de bruit sur les axes horizontaux. Le modèle MRU 10 est lui plutôt équilibré.

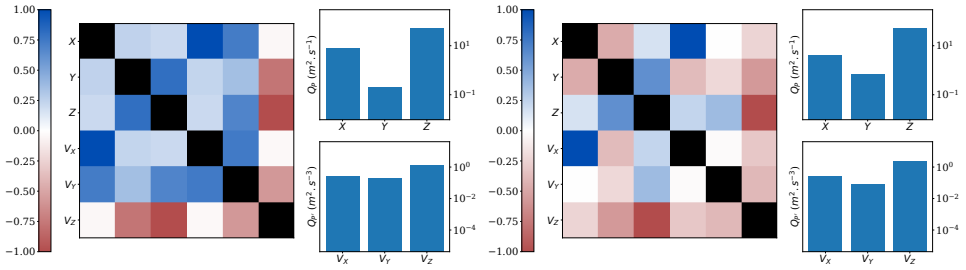


(a) MCU 01 (b) MCU 20

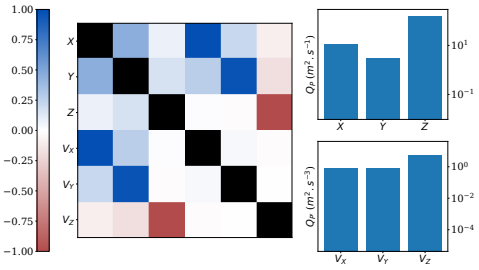
(c) MCU 02 (d) MCU 19

(e) MCU 03 (f) MCU 17

FIGURE 4.3 – Représentation des matrices de bruit de modèle pour les filtres MCU associés aux types de trajectoires 1, 2 et 3, à gauche pour $\gamma < 0$ (virage à droite) et à droite pour $\gamma > 0$ (virage à gauche), du virage le plus marqué en haut au moins marqué en bas.

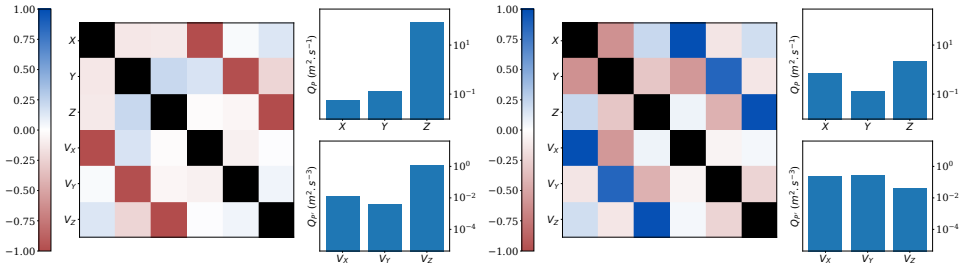


(a) MCU 04 (b) MCU 18

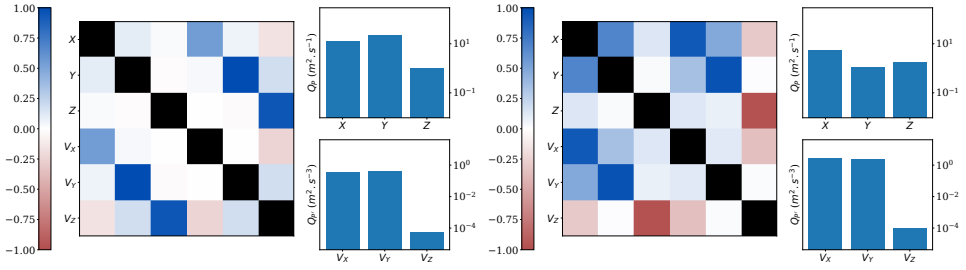


(c) MRU 14

FIGURE 4.4 – Représentation des matrices de bruit de modèle pour les filtres MCU associés au type de trajectoires 4, en haut à gauche pour $\gamma < 0$ (virage à droite) et en haut à droite pour $\gamma > 0$ (virage à gauche), la matrice du bas est celle du modèle MRU 14 qui fait le lien entre les deux.



(a) MRU 07 / MRU 15 (b) MRU 10



(c) MRU 06 (d) MRU 08

FIGURE 4.5 – Représentation des matrices de bruit de modèle pour les filtres MRU 07/15 associés au type de trajectoires 4, les filtres MRU 06 et 08 associés aux types de trajectoires 1, 2 et 3, et le filtre MRU 10 commun à tous les types de trajectoires.

4.5.4 Simplification de l'IMM

Puisque nous avons vu précédemment que certains modèles n'étaient quasiment jamais utilisés par l'IMM et que certains modèles MCU étaient en fait des modèles MRU, nous pouvons envisager de simplifier le filtre sans perte significative dans les performances de ce dernier. L'ensemble de résultats obtenus avec les IMM à l'issue de chacune des simplifications sont représentés dans le tableau 4.4.

La première simplification envisagée est le redressement des modèles MCU 06 à MCU 16 en modèles MRU. Cette opération ne présente pas de difficulté particulière puisque le modèle redressé peut conserver sa probabilité initiale, son rôle dans la matrice de transition ainsi que sa matrice de bruit de modèle. Comme on peut le voir dans le tableau 4.4, l'impact de cette simplification est, comme on pouvait s'y attendre, extrêmement faible.

L'autre opération de simplification envisagée est la suppression d'un modèle de l'IMM. Si la suppression de la matrice de bruit, de la probabilité initiale et éventuellement du taux de virage correspondants ne pose aucun problème, la modification de la matrice de transition s'avère plus compliquée. Nous proposons pour cela de trouver les poids $\tilde{\mathrm{W}}_{ij}$ dérivés des poids initiaux W_{ij} qui après régularisation par la fonction *softmax* ($\mathrm{T}_{ij} = \frac{e^{\mathrm{W}_{ij}}}{\sum_k e^{\mathrm{W}_{kj}}}$ et $\tilde{\mathrm{T}}_{ij} = \frac{e^{\tilde{\mathrm{W}}_{ij}}}{\sum_k e^{\tilde{\mathrm{W}}_{kj}}}$), permettant de vérifier la relation suivante pour la matrice de transition avant de supprimer la colonne et la ligne a :

$$\forall i, j \neq a, \quad \tilde{\mathrm{T}}_{ij} = \mathrm{T}_{ij} + \frac{\mathrm{T}_{ia}\mathrm{T}_{aj}}{1 - \mathrm{T}_{aa}}$$

Cette façon de faire revient dans le cas des modèles 07 et 15 à les fusionner si l'on supprime l'un des deux. Une des solutions consiste à effectuer la transformation suivante avant de supprimer la colonne et la ligne a :

$$\forall i, j \neq a, \quad \tilde{\mathrm{W}}_{ij} = \mathrm{W}_{ij} + \log\left(1 + \frac{e^{\mathrm{W}_{ia} + \mathrm{W}_{aj} - \mathrm{W}_{ij}}}{\sum_{k \neq a} e^{\mathrm{W}_{ka}}}\right)$$

Preuve :

$$
\begin{aligned}
\forall i, j \neq a, \quad \tilde{\mathrm{T}}_{ij} &= \frac{e^{\tilde{\mathrm{W}}_{ij}}}{\sum\limits_{k\neq a} e^{\tilde{\mathrm{W}}_{kj}}} \\
&= \frac{e^{\mathrm{W}_{ij}} + \frac{e^{\mathrm{W}_{ia}} e^{\mathrm{W}_{aj}}}{\sum\limits_{k\neq a} e^{\mathrm{W}_{ka}}}}{\sum\limits_{k\neq a} e^{\mathrm{W}_{kj}} + \frac{e^{\mathrm{W}_{aj}}}{\sum\limits_{k\neq a} e^{\mathrm{W}_{ka}}} \sum\limits_{k\neq a} e^{\mathrm{W}_{ka}}} \\
&= \frac{e^{\mathrm{W}_{ij}} + \frac{e^{\mathrm{W}_{ia}} e^{\mathrm{W}_{aj}}}{\sum\limits_{k\neq a} e^{\mathrm{W}_{ka}}}}{\sum\limits_{k} e^{\mathrm{W}_{kj}}} \\
&= \frac{e^{\mathrm{W}_{ij}}}{\sum\limits_{k} e^{\mathrm{W}_{kj}}} + \frac{e^{\mathrm{W}_{ia}}}{\sum\limits_{k\neq a} e^{\mathrm{W}_{ka}}} \frac{e^{\mathrm{W}_{aj}}}{\sum\limits_{k} e^{\mathrm{W}_{kj}}} \\
&= \frac{e^{\mathrm{W}_{ij}}}{\sum\limits_{k} e^{\mathrm{W}_{kj}}} + \frac{\frac{e^{\mathrm{W}_{ia}}}{\sum\limits_{k} e^{\mathrm{W}_{ka}}}}{\sum\limits_{k\neq a} \frac{e^{\mathrm{W}_{ka}}}{\sum\limits_{k} e^{\mathrm{W}_{ka}}}} \frac{e^{\mathrm{W}_{aj}}}{\sum\limits_{k} e^{\mathrm{W}_{kj}}} \\
&= \mathrm{T}_{ij} + \frac{\mathrm{T}_{ia}}{\sum\limits_{k\neq a} \mathrm{T}_{ka}} \mathrm{T}_{aj} \\
&= \mathrm{T}_{ij} + \frac{\mathrm{T}_{ia}\mathrm{T}_{aj}}{1-\mathrm{T}_{aa}}
\end{aligned}
$$

Pour déterminer quel modèle supprimer, on évalue les IMMs issus de chacune des amputations possibles, et on conserve celui obtenant la fonction de coût la plus basse. L'IMM ainsi obtenu est consolidé par une nouvelle phase d'apprentissage, et on recommence jusqu'à ce qu'il ne reste plus qu'un seul modèle. Le tableau 4.4 montre l'ordre de suppression des modèles, ainsi que les performances des IMMs élagués après consolidation. On constate que les premiers modèles supprimés sont le MRU 07 qui faisait doublon avec le MRU 15, ainsi que les MRU que nous avions qualifiés d'états transitoires précédemment (09, 11, 16, 13) à l'exception du MRU 12 dont la suppression ne se fera qu'après celles des MCU 02 et 17. Viennent ensuite le MCU 05 qui amenait une dissymétrie entre virages à gauche et virages à droite, puis le MRU 10 qui n'était pas discriminant entre le type 4 et les autres types de trajectoires. Jusque là, aucune des 10 suppressions n'avait entraîné une perte significative de performances dans l'IMM, le meilleur filtre au regard de la fonction de coût arrivant après la quatrième amputation. Au-delà, avec la suppression des MCU 03 et 19 qui augmentent chacun l'erreur en position de plus d'1m, presque chaque élagage provoque une perte de précision non-négligeable sur au moins l'une des métriques. Le dernier modèle restant est le MRU 14 qui avait une matrice de bruit similaire à celle apprise pour un filtre de Kalman simple avec modèle MRU, on retrouve d'ailleurs des

performances très similaires sur chacune des métriques.

Opération	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	Loss
IMM initial	44.46	40.35	2.55	10.65	14.525
MCUs 06 à 16 redressés	44.56	40.33	2.57	10.64	14.572
MRU 09 supprimé	44.15	40.16	2.57	10.32	14.428
MRU 11 supprimé	44.19	**40.12**	2.57	10.34	14.424
MRU 16 supprimé	44.13	40.14	2.57	10.32	14.420
MRU 07 supprimé	44.12	40.20	**2.57**	**10.30**	**14.406**
MRU 13 supprimé	**44.08**	40.18	2.57	10.31	14.407
MCU 02 supprimé	44.33	40.24	2.58	10.39	14.440
MCU 17 supprimé	44.50	40.23	2.59	10.46	14.465
MRU 12 supprimé	44.77	40.41	2.60	10.51	14.516
MCU 05 supprimé	44.68	40.42	2.60	10.54	14.550
MRU 10 supprimé	44.83	40.84	2.59	10.46	14.656
MCU 03 supprimé	46.00	41.07	2.62	10.72	14.932
MCU 19 supprimé	47.75	41.09	2.68	11.05	15.177
MRU 08 supprimé	48.39	40.93	2.80	11.73	15.199
MRU 15 supprimé	49.18	41.87	2.85	11.73	15.491
MCU 20 supprimé	49.52	44.78	2.79	12.49	15.968
MCU 04 supprimé	48.54	45.31	2.77	12.21	15.931
MRU 06 supprimé	57.29	43.91	3.75	15.49	17.125
MCU 01 supprimé	59.33	46.34	3.57	15.84	18.224
MCU 18 supprimé	59.73	46.57	4.71	17.40	18.460
FK	59.62	46.69	4.77	17.51	18.467

TABLEAU 4.4 – Récapitulatif des opérations de simplifications effectuées sur l'IMM et leur impact sur les différentes métriques exprimées sous la forme de leur REQM ainsi que sur la fonction de coût.

4.5.5 Conclusions sur l'IMM à modèles MCU

Dans cette section, nous avons étudié en détail le fonctionnement de l'IMM à 20 modèles MCU obtenu par optimisation bayésienne précédemment. Nous avons pu identifier relativement clairement le rôle de la plupart des modèles et mis en évidence une forme de classification opérée par l'IMM visible dans l'utilisation qui est faite de ceux-ci.

L'un des avantages à n'utiliser que des modèles MCU est que le nombre d'hyper-paramètres nécessaires à l'implémentation du filtre est réduit à un. Au cours de l'apprentissage, certains modèles MCU sont capables d'approcher un taux de virage suffisamment proche de zéro pour être considérés comme des modèles MRU et leur spécialisation peut donc se faire automatiquement. Nous avons aussi pu par la suite simplifier l'IMM jusqu'à en retirer la moitié des modèles sans perte significative de performance.

4.6 Intégration d'une cellule LSTM

Comme nous l'avons vu dans le chapitre précédent, l'intégration d'une cellule LSTM permet aux filtres d'opérer une forme de classification et de garder en mémoire certaines caractéristiques de l'appareil suivi lui permettant d'améliorer ses performances de pistage. Dans cette section nous proposerons un algorithme basé sur l'IMM intégrant un réseau LSTM.

4.6.1 Architectures

Première version

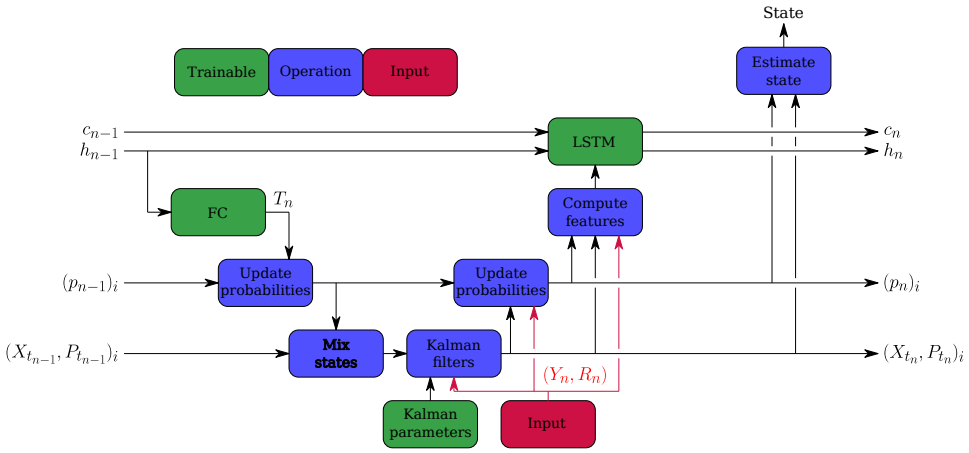


FIGURE 4.6 – Schéma de l'IMM à LSTM.

La première version de l'IMM à cellule LSTM proposée et représentée en Figure 4.6 intègre un réseau prenant un certain nombre de grandeurs calculées à la suite des mises à jours des filtres de Kalman qui composent l'IMM et de leurs probabilités associées. La taille de la cellule, et donc de la quantité d'information que le filtre pourra stocker de cette manière, est un hyper-paramètre du filtre. La

liste des grandeurs utilisées en entrée du LSTM au temps t_n est la suivante :

la vitesse au sol :

$$\mathrm{V}_{h,n} = \sum_m (p_n)_m ((\mathrm{V}_{\mathrm{X},n})^2_m + (\mathrm{V}_{\mathrm{Y},n})^2_m)^{\frac{1}{2}} \tag{4.3}$$

la vitesse verticale :

$$\mathrm{V}_{v,n} = \sum_m (p_n)_m (\mathrm{V}_{\mathrm{Z},n})_m \tag{4.4}$$

le taux de virage [1] :

$$\gamma_n = \sum_{m \in \mathrm{CT}} (p_n)_m \gamma_m + \sum_{m \in \mathrm{CA} \cup \mathrm{Singer}} (p_n)_m (\gamma_n)_m \tag{4.5}$$

le gain en position (matrice 3×3) :

$$\mathrm{K}_{\mathrm{P},n} = \sum_m (p_n)_m \mathrm{H}_m . (\mathrm{K}_n)_m \tag{4.6}$$

la correction en position (vecteur 3) :

$$\Delta_{\mathrm{P},n} = \sum_m (p_n)_m \mathrm{H}_m . (\mathrm{K}_n)_m . (\mathrm{Z}_n)_m \tag{4.7}$$

le carré de l'innovation normalisée (vecteur 3) :

$$\mathrm{I}_n = \sum_m (p_n)_m \frac{(\mathrm{Z}_n)^2_m}{diag(\mathrm{H}_m . (\mathrm{P}_n)_m . \mathrm{H}^{\mathrm{T}}_m)} \tag{4.8}$$

La sortie de cette cellule LSTM est utilisée par une couche de neurones de taille k^2 où k est le nombre de modèles. Après avoir été remodelée en une matrice $(k \times k)$ et régularisée par une fonction *softmax*, la sortie de cette couche de neurones est utilisée comme matrice de transition T_n au temps t_n pour la mise à jour des probabilités de l'IMM. C'est par cet unique biais que le réseau de neurones pilote l'utilisation des différents filtres.

Seconde version

Dans cette seconde version, une autre couche de neurones se base sur l'expression de la cellule LSTM et permet, après régularisation par une fonction *softmax*, d'estimer une classification explicite de la cible. Dans notre cas, les classes à déterminer sont les quatre types de trajectoires utilisés à la génération des jeux de données. Une représentation de ce filtre est présentée en Figure 4.7.

4.6.2 Expériences numériques

Les hyper-paramètres de l'IMM à LSTM ont été fixés grâce aux connaissances obtenues lors de l'apprentissage des filtres vu précédemment. Le choix

1. les taux de virage des modèles MUA et Singer sont calculés par la formule $\gamma = \frac{\mathrm{A_Y V_X - A_X V_Y}}{\mathrm{V_X^2 + V_Y^2}}$, ceux des modèles MRU et arrêt sont nuls, ceux des modèles MCU sont constants

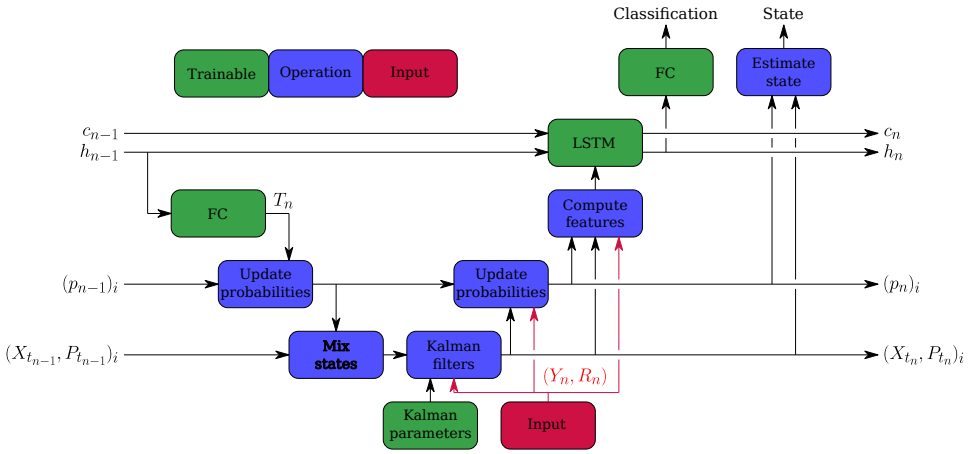


FIGURE 4.7 – Schéma de l'IMM à LSTM.

des modèles pour l'IMM a été de n'utiliser que des modèles MCU en 20 exemplaires comme précédemment, et la taille de la cellule LSTM a été fixée à 10 comme pour le nombre de matrices d'adaptation du NNAKF.

Si aucun changement n'est nécessaire pour entraîner la première version de l'algorithme puisque ses entrées et sorties sont parfaitement comparables à celles des filtres étudiés précédemment, ce n'est pas le cas de la seconde. En effet, l'expression de la classification doit être prise en compte dans l'évaluation de la fonction de coût pour que la deuxième couche de neurones correspondant à cette tâche puisse être entraînée. Nous introduisons donc le terme d'entropie croisée $\mathscr{H}_{classif}$ défini ci-dessous faisant intervenir l'expression attendue de la classification en encodage *one-hot* c_i et la probabilité $\hat{c}_i$ exprimée par le filtre de la cible d'appartenir à la classe i. La fonction de coût finale $\mathscr{H}$ est une pondération entre ce terme et la fonction de coût précédemment utilisée $\mathscr{H}_{track}$ dépendant d'un coefficient λ.

$$\mathscr{H}_{track} = \frac{1}{2}\|\mathrm{H}_n\mathrm{X}^+_{t_n} - \mathrm{Y}_n\|^2_{\mathrm{P}^+_{t_n}} + \frac{1}{2}\log|\mathrm{P}^+_{t_n}| \tag{4.9}$$

$$\mathscr{H}_{classif} = -\sum_i c_i \log(\hat{c}_i) \tag{4.10}$$

$$\mathscr{H} = \lambda\mathscr{H}_{track} + (1-\lambda)\mathscr{H}_{classif} \tag{4.11}$$

Dans la Figure 4.8, nous étudions l'influence du paramètre λ. Pour chacune des valeurs considérées, cinq entraînements ont été réalisés. La figure représente en trait plein les valeurs médianes et en pointillés les valeurs extrêmes obtenues sur ces cinq entraînements pour les deux termes qui composent la fonction de coût. On peut constater que l'un des termes croît lorsque l'autre décroît, il est en effet intuitif pour chacun des deux termes d'être plus bas lorsqu'on lui accorde plus d'importance dans la fonction de coût totale. On peut aussi remarquer qu'il existe un bon compromis aux alentours de $\lambda = \frac{1}{2}$ pour lequel les

deux termes ont des valeurs proches de leur minimum. Dans la suite nous la fixons donc à cette valeur.

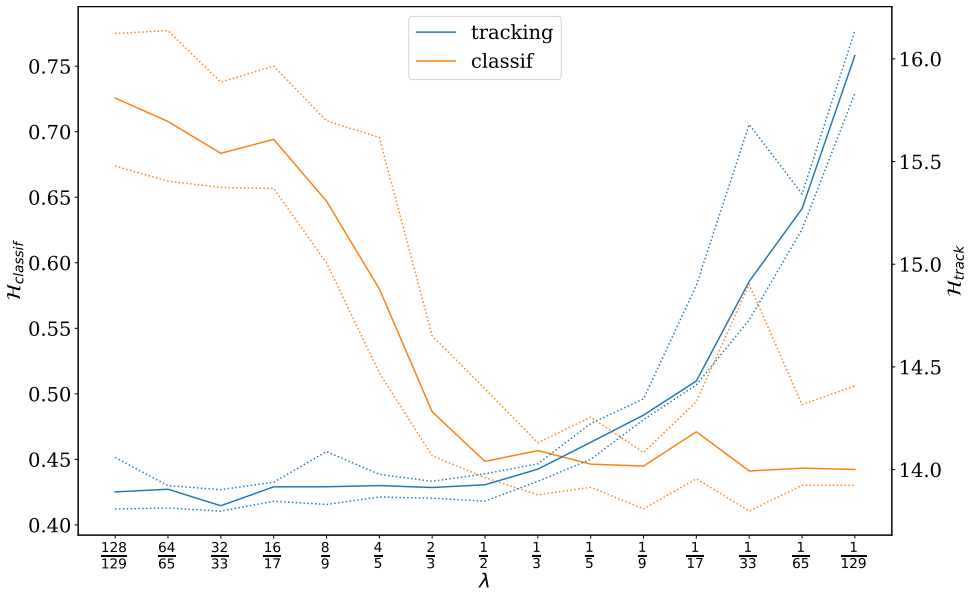


FIGURE 4.8 – Évolution des valeurs des deux termes de la fonction de coût après entraînement en fonction du paramètre λ

Le tableau 4.5 donne les REQM des différents IMM sur les quatre métriques, ainsi que la valeur de la fonction de coût restreinte à la problématique du pistage ($\mathscr{H}_{track}$) et la précision de la classification en fin de trajectoire. On constate que l'intégration de la cellule LSTM améliore la précision du pistage sur toutes les métriques, mais l'écart est moins important que lorsque nous étions passés du filtre de Kalman simple au NNAKF. L'ajout de la classification ne dégrade que très peu la précision du pistage et permet une discrimination plus que correcte du type de trajectoire. La matrice de confusion liée à cette classification est représentée dans la figure 4.9. On peut notamment y constater que les trajectoires du type 4 sont les plus faciles à discriminer, ce qui rejoint les observations faites sur les modèles utilisés par l'IMM sans LSTM en fonction du type de trajectoire.

Nous pouvons aussi nous intéresser à l'évolution de la classification au cours du pistage pour simuler une utilisation en temps réel et observer son temps de réaction. La précision (*precision*) mesure parmi les trajectoires classées dans chaque catégorie quelle part d'entre elles appartiennent bien à cette classe. Le rappel (*recall*) mesure parmi les trajectoires de chaque catégorie quelle part a été correctement classée. Le F1-score est un compromis entre ces deux mesures, $\text{F1} = \frac{2\times recall\times precision}{recall+precision}$. Dans la figure 4.10, nous avons représenté l'évolution de ces trois indicateurs au cours du temps pour chacune des classes. Nous pouvons y voir que les scores s'améliorent fortement durant les deux premières minutes et commencent vraiment à stagner après cinq minutes.

Filtre	Position (m)	Altitude (m)	Vitesse ($m.s^{-1}$)	Cap (°)	$\mathscr{H}_{track}$	Classification (%)
Oracle	28.26	13.66	2.50	3.37	NA	NA
FK	59.62	46.69	4.77	17.51	18.467	NA
Castella	54.11	46.62	3.81	14.29	17.507	NA
NNAKF MRU LSTM 10M	51.05	42.82	3.49	12.25	16.182	NA
IMM 20MCU	44.46	40.35	2.55	10.65	14.549	NA
IMM-LSTM 10u 20MCU	**42.99**	**39.38**	**2.47**	**8.54**	**13.822**	NA
IMM-LSTM 10u 20MCU + classif	43.17	39.74	2.49	8.65	13.931	**83.4**

TABLEAU 4.5 – REQM sur différentes métriques et fonction de coût du pistage des IMM à 20 modèles MCU présentés dans ce chapitre avec rappel de ces valeurs pour les filtres à un seul modèle et résultat de la classification.

4.6.3 Conclusions sur l'IMM à LSTM

Dans cette section, nous avons proposé d'intégrer une cellule LSTM à un IMM à 20 modèles MCU et de lui donner la possibilité de piloter indirectement le filtre par le biais de la matrice de transition dans un premier temps, puis d'estimer une classification explicite du type de trajectoire suivie dans un second temps. Le filtre ainsi entraîné montre un gain en performances sur chacune des métriques considérées, bien que celui-ci soit moins important que pour l'ajout de la cellule LSTM au filtre de Kalman simple menant au NNAKF. L'ajout de la classification explicite issue de la cellule LSTM ne permet certes pas d'améliorer la précision du pistage, mais ne la dégrade pas vraiment non plus, elle permet par contre une discrimination correcte entre les types de trajectoires du problème considéré. La partie classifieur telle qu'elle est proposée ici peut fournir des caractéristiques propres à la cible en temps réel aux opérateurs chargés de la surveillance aérienne, quelques minutes d'observation sont suffisantes pour obtenir une classification stable.

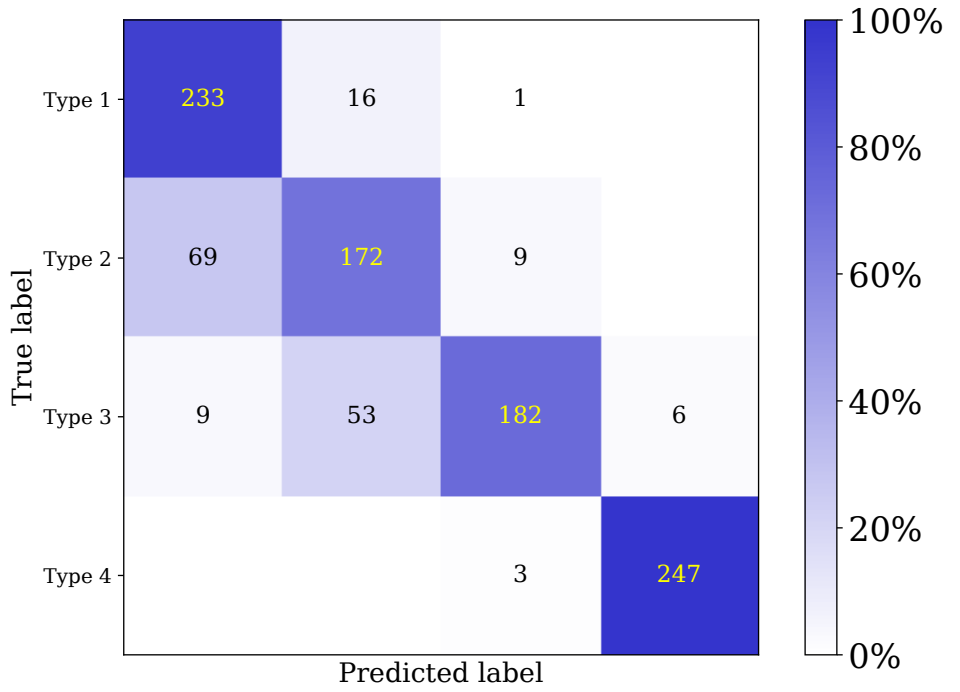


FIGURE 4.9 – Matrice de confusion de la classification issue de l'IMM avec LSTM à 20 modèles MCU en fin de trajectoire.

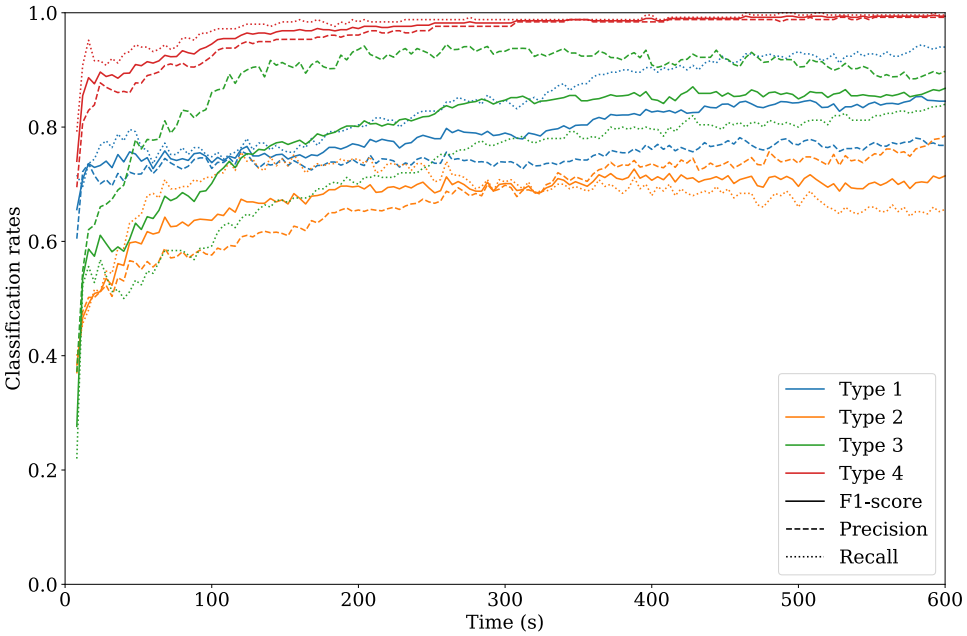


FIGURE 4.10 – Évolution de la précision, du rappel et du F1-score des différents types de trajectoire

4.7 Tests de robustesse

4.7.1 Expériences numériques

Robustesse au bruit

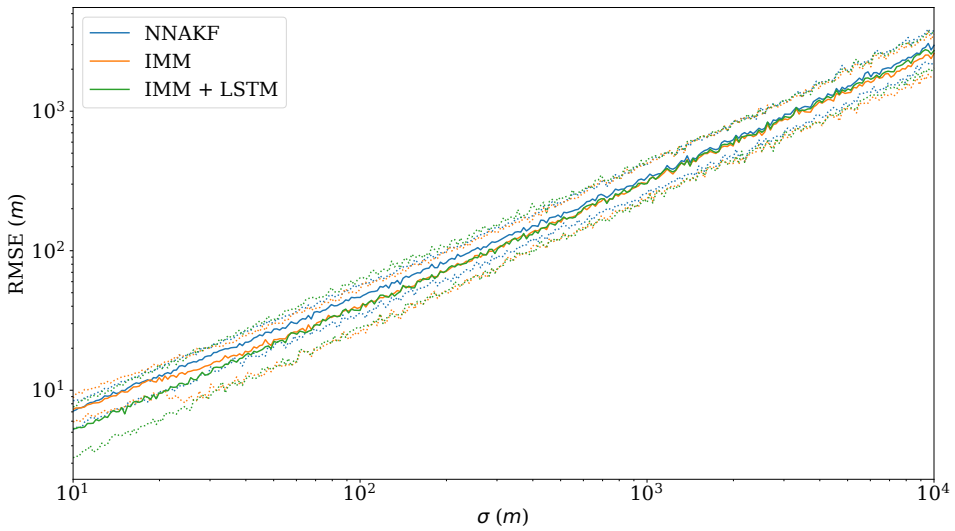


FIGURE 4.11 – Évolution de la REQM en position des différents filtres en fonction de l'intensité du bruit de mesure. Les courbes en trait plein représentent les médianes et celles en pointillés les premier et dernier déciles.

Les filtres IMM à 20 MCU avec et sans cellule LSTM ont été soumis au même test de robustesse au bruit que dans la section 3.4.4 et comparés aux résultats déjà obtenus avec le NNAKF. On peut remarquer sur la figure 4.11 que la REQM peut encore ici être assez bien reliée au niveau de bruit σ par une loi de puissance REQM = $e^{\beta}\sigma^{\alpha}$. Comme précédemment, les tableaux 4.6, 4.7 et 4.8 nous permettent de voir si l'un des paramètres du scénario de test a une influence sur les performances des filtres.

Dans le tableau 4.6, on peut constater que contrairement à tous les autres filtres étudiés précédemment, l'IMM avec LSTM présente des résultats différents si la trajectoire est verticale ou horizontale. En effet, contrairement à la cellule LSTM du NNAKF qui ne prenait en entrée que les innovations normalisées, celle de l'IMM utilise bien plus d'informations, y compris la vitesse verticale. Étant donné qu'aucune des trajectoires du jeu de données ne prend une direction purement verticale, le comportement de ce filtre est naturellement imprévisible sur ces trajectoires et il n'est donc pas surprenant que sa précision soit dégradée par rapport aux trajectoires horizontales.

Dans le tableau 4.7, on peut remarquer que, comme pour la direction, seuls les résultats de l'IMM avec LSTM sont significativement affectés par la vitesse de la trajectoire. Là encore, ce comportement peut s'expliquer par la composi-

Filtre	Direction	α	e^{β} (m)	R^2
NNAKF	$\vec{e}_X$	0.869	0.86	0.987
	$-\vec{e}_X$	0.870	0.85	0.987
	$\vec{e}_Y$	0.868	0.86	0.987
	$-\vec{e}_Y$	0.870	0.86	0.987
	$\vec{e}_Z$	0.872	0.85	0.987
	$-\vec{e}_Z$	0.870	0.86	0.987
IMM	$\vec{e}_X$	0.871	0.79	0.975
	$-\vec{e}_X$	0.872	0.79	0.975
	$\vec{e}_Y$	0.874	0.77	0.975
	$-\vec{e}_Y$	0.874	0.77	0.974
	$\vec{e}_Z$	0.869	0.77	0.981
	$-\vec{e}_Z$	0.867	0.77	0.981
IMM+LSTM	$\vec{e}_X$	0.916	0.58	0.974
	$-\vec{e}_X$	0.920	0.57	0.975
	$\vec{e}_Y$	0.918	0.57	0.975
	$-\vec{e}_Y$	0.920	0.57	0.974
	$\vec{e}_Z$	0.868	0.82	0.982
	$-\vec{e}_Z$	0.892	0.62	0.986

TABLEAU 4.6 –
Coefficients de la régression linéaire de la REQM en fonction du niveau de bruit sur une double échelle logarithmique ainsi que le R^2 de celle-ci pour les différents filtres et selon les directions possibles de la trajectoire.

Filtre	Vitesse $(m.s^{-1})$	α	e^{β} (m)	R^2
NNAKF	1	0.868	0.87	0.987
	10	0.869	0.86	0.987
	100	0.870	0.85	0.987
	1000	0.872	0.85	0.986
IMM	1	0.865	0.78	0.982
	10	0.865	0.78	0.981
	100	0.858	0.86	0.979
	1000	0.896	0.69	0.968
IMM+LSTM	1	0.946	0.43	0.988
	10	0.919	0.53	0.984
	100	0.908	0.60	0.982
	1000	0.849	1.10	0.984

TABLEAU 4.7 –
Coefficients et R^2 selon les vitesses possibles de la trajectoire.

Filtre	Axe bruité	α	e^{β} (m)	R^2
NNAKF	R_{XX}	0.854	0.99	0.991
	R_{YY}	0.855	0.99	0.991
	R_{ZZ}	0.901	0.64	0.987
IMM	R_{XX}	0.853	0.92	0.984
	R_{YY}	0.859	0.86	0.982
	R_{ZZ}	0.902	0.59	0.971
IMM+LSTM	R_{XX}	0.889	0.68	0.981
	R_{YY}	0.898	0.65	0.981
	R_{ZZ}	0.930	0.54	0.970

TABLEAU 4.8 –
Coefficients et R^2 selon les directions possibles du bruitage.

tion du jeu de données d'entraînement. Les trajectoires les plus rapides qui s'y trouvent sont celles de type 4 qui sont limitées à $250 m.s^{-1}$, des trajectoires allant à $1000 m.s^{-1}$ sont donc inédites pour le filtre.

Enfin, le tableau 4.8 montre une tendance similaire pour les IMM que pour le NNAKF dans la dépendance de leurs résultats en fonction de la direction du bruit, avec des résultats similaires pour les axes horizontaux et une erreur moindre lorsque l'axe vertical est bruité.

Robustesse pour une trajectoire uniformément accélérée

Les IMMs ont ici été soumis au test présenté dans la section 3.4.4. Les évolutions des erreurs en fonction de l'accélération sont représentées sur la figure 4.12. Le comportement des IMM est comparable à celui des autres filtres, à savoir une phase où l'erreur augmente peu, puis une autre où elle décolle car le bruit de modèle ne peut plus suivre.

Il est clair que pour les trajectoires horizontales, le NNAKF s'en sort largement mieux à forte accélération que les IMMs, ce n'est pas vrai pour les trajectoires verticales pour lesquelles c'est l'IMM avec LSTM qui a un net avantage. Un autre fait notable est la présence de valeurs d'erreur ponctuellement plus fortes

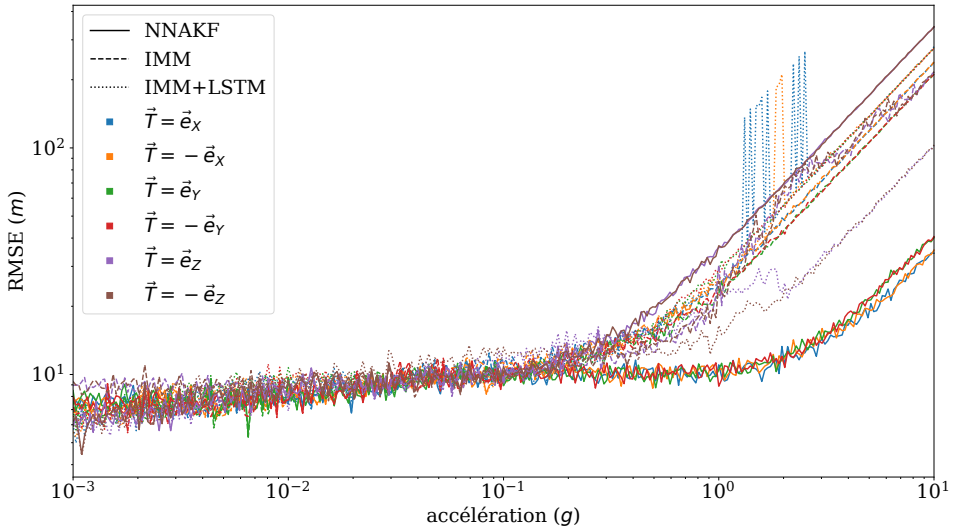


FIGURE 4.12 – Évolution de la REQM en position des différents filtres en fonction de l'accélération donnée aux trajectoires dans la direction $\vec{T}$.

d'un facteur environ 4 par rapport à la tendance pour l'IMM avec LSTM, ce qui met en évidence une possible instabilité du filtre.

Robustesse pour une trajectoire circulaire uniforme

Dans ce nouveau test, les trajectoires suivies sont des trajectoires circulaires horizontales pouvant prendre deux valeurs de vitesse différentes ($100 m.s^{-1}$ ou $1000 m.s^{-1}$) et aux taux de virages variant de $\frac{\pi}{150} rad.s^{-1}$ à $\frac{\pi}{20} rad.s^{-1}$. L'erreur est mesurée sur la position dans le plan horizontal.

Pour une vitesse de $100 m.s^{-1}$ et jusqu'à un taux de virage d'environ $0.1 rad.s^{-1}$, les résultats du NNAKF et des IMMs sont relativement stables avec un avantage constant pour les IMMs. Passé ce niveau de taux de virage, l'erreur des IMMs dépasse largement celle du NNAKF. Nous avions déjà vu que les taux de virage maximum appris pour les modèles MCU des IMMs sur notre jeu de données étaient d'environ $0.08 rad.s^{-1}$, ce qui est cohérent avec ce "seuil de décrochage" observé.

Pour une vitesse de $1000 m.s^{-1}$, l'erreur du NNAKF est plus conséquente et augmente clairement mais régulièrement avec le taux de virage. Les IMMs ont eux un comportement bien plus erratique et difficilement comparable à celui du NNAKF, on peut toutefois y observer pour chaque courbe un creux profond et étroit aux abords de la valeur $0.08 rad.s^{-1}$ qui correspond aux taux de virages des modèles MCU extrêmes appris par les filtres. Passés ce taux de virage critique, l'erreur décolle comme pour les trajectoires à $100 m.s^{-1}$.

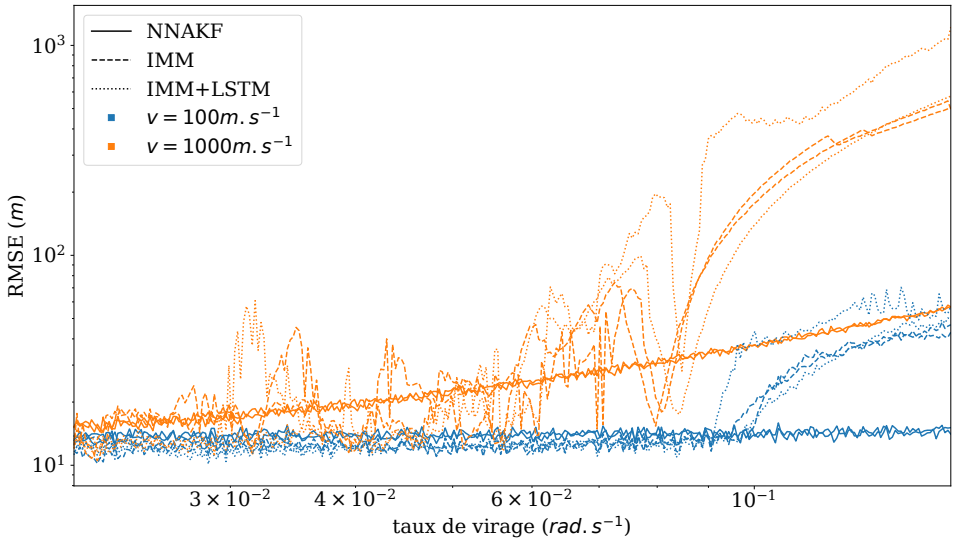


FIGURE 4.13 – Évolution de la REQM en position des différents filtres en fonction du taux de virage donné aux trajectoires pour deux valeurs de vitesse différentes. Les deux sens de rotation sont représentés, d'où la présence à chaque fois de deux courbes de même couleur et style. Nous n'avons pas jugé nécessaire de les distinguer.

4.7.2 Conclusion

Bien que les résultats de l'IMM soient très bons lorsque mesurés sur un jeu de données composé de trajectoires proches de celles utilisées lors de l'entraînement, il semble qu'ils peuvent très vite se dégrader pour des trajectoires trop différentes de celles déjà rencontrées. Ce que ces filtres ont gagné en performance en réglant des modèles avec des taux de virages spécifiquement adaptés au jeu de données considéré est perdu en robustesse, contrairement au NNAKF qui, en jouant sur les matrices de bruit, permet une adaptation plus générale. Il est donc à noter que pour ceux-ci, encore plus que pour le NNAKF ou les filtres classiques, le choix du jeu de données d'apprentissage est crucial. Toutes les caractéristiques possibles des trajectoires qu'ils pourraient être amenés à rencontrer lors de leur utilisation réelle, tant pour l'accélération, les virages, la vitesse, et même la direction, doivent être représentées dans le jeu de données d'entraînement pour éviter de graves instabilités.

4.8 Conclusion

Dans ce chapitre, nous avons étudié les possibilités d'apprentissage de filtres à modèles multiples. L'un des modèles considérés, le modèle circulaire uniforme (MCU) s'est avéré être largement utilisé par le filtre réglé par optimisation bayesienne. Une analyse plus approfondie a permis de montrer que même en utilisant ce modèle, le filtre était capable au cours de l'apprentissage d'approcher

suffisamment son taux de virage de zéro pour en faire un modèle rectiligne uniforme (MRU), cas limite de celui-ci. Une version modifiée de l'IMM intégrant un réseau de neurones a ensuite été proposée, le réseau en question a la capacité de piloter au besoin le filtre par le biais de sa matrice de transition. L'IMM puis le réseau de neurones ont permis d'améliorer de façon significative les performances de filtrage par rapport aux propositions précédentes. De plus, la mémoire du LSTM permet une forme de classification implicite de la cible qu'il est assez facile d'expliciter à l'aide d'un autre réseau de neurones.

Cependant, plusieurs points méritent notre attention. Tout d'abord le coût calculatoire des filtres à modèles multiples est bien plus grand que celui des filtres à modèle unique, à la fois pour la phase d'apprentissage et pour son utilisation réelle. Ensuite, la robustesse de ces filtres montre certains défauts. Malgré les précautions prises pour éviter les problèmes de sur-apprentissage (jeu de données de validation pour arrêter l'entraînement à temps, faible nombre de paramètres entraînables), dès que la cible à pister a une trajectoire un peu trop différente de celles qui lui ont servi à s'entraîner la précision du filtre diminue considérablement. Tout semble indiquer que ce filtre a plus de mal que le NNAKF à extrapoler le comportement à avoir pour de nouvelles trajectoires. Cette observation rejoint l'hypothèse que plus un filtre sera complexe, plus il aura de degrés de liberté (ici matrices de bruit, taux de virages, cellule LSTM et matrice de transition), et plus il se spécialisera sur les particularités du jeu de données d'entraînement qui lui aura été fourni. Il est donc très important pour utiliser un tel filtre de se doter au préalable d'un jeu de données adapté aux cibles que l'on souhaitera pister par la suite, et comportant un panel exhaustif des situations auxquelles il pourra être confronté.

4.9 Références

BARRAU, A. et S. BONNABEL. 2016, «The invariant extended kalman filter as a stable observer», *IEEE Transactions on Automatic Control*, vol. 62, nº 4, p. 1797–1812. 68

BLOM, H. A. et Y. BAR-SHALOM. 1988, «The interacting multiple model algorithm for systems with markovian switching coefficients», *IEEE transactions on Automatic Control*, vol. 33, nº 8, p. 780–783. 68

GELB, A. 1974, *Applied Optimal Estimation*, MIT press, 182-190 p.. 68

PILTÉ, M., S. BONNABEL et F. BARBARESCO. 2017, «Tracking the frenet-serret frame associated to a highly maneuvering target in 3d», dans *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, p. 1969–1974. 68

Chapitre 5

Conclusions

Cette thèse a permis d'explorer différentes possibilités d'amélioration des algorithmes de filtrage actuels à l'aide de réseaux de neurones. Nous y avons traité à la fois d'algorithmes de filtrage, mais aussi de classification automatique d'aéronefs. La problématique de la classification automatique des cibles a été abordée de deux façons distinctes.

D'abord par l'utilisation de réseaux de neurones profonds s'appuyant sur des grandeurs cinétiques pouvant être obtenues à la suite d'un filtrage, bien que dans le Chapitre 2 le jeu de données soit issu d'un lissage de transmissions ADS-B. Cette méthode permet de distinguer sans trop d'erreur les différentes grandes catégories d'appareils sans toutefois pouvoir efficacement discriminer les cibles avec plus de détail, il n'est par exemple pas facile de distinguer un avion de ligne bimoteur d'un quadrimoteur en revanche on pourra plutôt bien reconnaître s'il s'agit d'un hélicoptère, un avion de chasse ou un avion de ligne. De plus cette méthode peut être facilement ajoutée à n'importe quel algorithme de pistage déjà existant.

La classification a ensuite été traitée par l'utilisation de petits réseaux de neurones directement au sein de l'algorithme de pistage comme dans la deuxième version de l'IMM du Chapitre 4. Contrairement à la méthode précédente, celle-ci est étroitement liée à l'algorithme de filtrage auquel elle est greffée et ne peut donc pas être ajoutée à un algorithme déjà existant. Les résultats obtenus sur les trajectoires simulées sont plutôt encourageant, mais malheureusement aucun test n'a pu être effectué sur trajectoires réelles faute de données adaptées. Cette méthode de classification, bien que fortement liée à l'algorithme de filtrage, n'induit pas de boucle de rétroaction pouvant mener à un cercle vicieux (mauvaise classification entraînant un mauvais pistage entraînant une encore plus mauvaise classification et ainsi de suite) et ne perturbe donc pas la stabilité de l'algorithme de pistage.

La problématique du filtrage a elle aussi été abordée de deux façons distinctes.

D'abord sous l'angle du filtrage adaptatif, dans le Chapitre 3 le filtre proposé et baptisé NNAKF est capable d'ajuster dynamiquement ses paramètres de bruit de modèle en fonction de la situation. Cette méthode permet d'augmenter de façon significative la précision de l'algorithme de pistage sans pour autant trop augmenter les temps de calcul. De plus cette adaptation semble particulièrement robuste, sur toutes les trajectoires testées sa précision semble à minima équivalente à celle du filtre privé du module d'adaptation. Son utilisation ne présente donc aucun désavantage significatif pour un avantage non négligeable sur la qualité de l'algorithme de pistage. Il pourrait être intéressant pour la suite de tester ses effets sur des filtres dotés de modèles plus compliqués, comme par exemple l'IEKF à paramètres de Frenet constants.

Nous nous sommes ensuite intéressé au filtrage à modèles multiples avec notamment l'utilisation d'un grand nombre de modèles circulaires uniformes. Cette méthode permet une précision encore accrue du filtrage par rapport au NNAKF mais nécessite de plus grands temps de calcul. De plus les tests de robustesse ont montré que ce type de filtre perd considérablement en précision lorsque les trajectoires diffèrent trop par rapport à celles utilisées au cours de l'entraînement. En conséquence, son utilisation ne peut être recommandée que si l'utilisateur dispose d'un jeu de données d'entraînement représentant assez fidèlement les différentes trajectoires que l'algorithme pourra rencontrer au cours de son utilisation. La meilleur piste pour constituer un tel jeu de données serait l'utilisation de données réelles jumelée d'une méthode d'augmentation des données permettant de simuler des trajectoires variées et réalistes.

Annexe A

Annexes

Sommaire

A.1 Simulation de trajectoires

A.1.1 Principe général

Toutes les trajectoires générées sont d'abord créées lisses et à un pas de temps court δt. Pour ce faire elles sont le résultat de l'intégration de commandes de vol Γ elles-mêmes générées aléatoirement. Les trajectoires obtenues sont caractérisées par les positions et vecteurs vitesses de la cible au cours du temps, obtenue pas à pas suivant une relation de la forme :

$$(\mathrm{X},\mathrm{Y},\mathrm{Z},\mathrm{V_X},\mathrm{V_Y},\mathrm{V_Z})_{i+1} = f((\mathrm{X},\mathrm{Y},\mathrm{Z},\mathrm{V_X},\mathrm{V_Y},\mathrm{V_Z})_i,\Gamma_i,\delta t) \tag{A.1}$$

Les plots à fournir en entrée au filtres sont ensuite générés à partir de ces trajectoires qui serviront de référence pour l'apprentissage. Ils sont le résultat de bruitage des positions en coordonnées cartésienne ou transformées en coordonnées polaires pour simuler des détections radar, à des dates tirées aléatoirement ou à pas de temps régulier.

A.1.2 Le choix des commandes de vol

Au cours de ces travaux un jeu de trois commandes simples a été choisi pour générer les trajectoire. Il s'agit de l'accélération longitudinale dans le plan horizontal A, le taux de virage γ et la vitesse verticale $\mathrm{V_Z}$. De plus une fonction $\mathrm{B}_{v_{min},v_{max}}$ a été définie dans le but de borner la vitesse au sol entre deux valeurs limites v_{min} et v_{max} :

$$\mathrm{B}_{v_{min},v_{max}} : \tilde{v} \mapsto v = v_{min} + (v_{max} - v_{min})\sigma(2\frac{2\tilde{v} - (v_{min} + v_{max})}{v_{max} - v_{min}}) \tag{A.2}$$

Il s'agit d'une transformation de la fonction sigmoïde σ aillant pour limite en $-\infty$ (resp. $+\infty$) la vitesse limite v_{min} (resp. v_{max}) et l'identité pour tangente au voisinage de la vitesse moyenne $\frac{v_{min}+v_{max}}{2}$. En définissant θ le cap de la cible, V sa vitesse au sol bornée et $\tilde{\mathrm{V}}$ sa vitesse au sol non bornée, on peut écrire les étapes de création de la trajectoire comme suit :

$$X_{i+1} = X_i + \delta t V_{Xi} \tag{A.3}$$
$$Y_{i+1} = Y_i + \delta t V_{Yi} \tag{A.4}$$
$$Z_{i+1} = Z_i + \delta t V_{Zi} \tag{A.5}$$
$$\theta_{i+1} = \theta_i + \delta t \gamma_i \ [2\pi] \tag{A.6}$$
$$\tilde{V}_{i+1} = \tilde{V}_i + \delta t A_i \tag{A.7}$$
$$V_{i+1} = B_{v_{min},v_{max}}(\tilde{V}_{i+1}) \tag{A.8}$$
$$V_{Xi+1} = V_{i+1} \cos\theta_{i+1} \tag{A.9}$$
$$V_{Yi+1} = V_{i+1} \sin\theta_{i+1} \tag{A.10}$$
$$\tag{A.11}$$

La position initiale est fixée à l'origine du repère ($X_0 = Y_0 = Z_0 = 0$), le cap initial est tiré de façon uniforme entre 0 et 2π, et la vitesse non bornée initiale est tirée de façon uniforme entre v_{min} et v_{max}. V_Z est entièrement définie par la commande de vol correspondante.

A.1.3 Génération des commandes de vol

Toutes les commandes de vol sont générées de la même façon. On part d'un bruit blanc gaussien dont on pourra régler l'amplitude σ. Il est ensuite lissé par un filtre passe-bas dont on définira la fréquence caractéristique. Ce lissage se fait par convolution avec la fonction :

$$f_\nu : u \mapsto \frac{1}{1+(\nu u)^2} \tag{A.12}$$

Le spectre du signal filtré décroît exponentiellement avec la fréquence. Afin de conserver l'amplitude du signal, f_ν est normalisée par $\|f_\nu\|_2$. La dernière étape consiste à appliquer la fonction suivante au signal :

$$c_M : x \mapsto \begin{cases} x+M & \text{si } x < -M \\ 0 & \text{si } -M < x < M \\ x-M & \text{si } M < x \end{cases} \tag{A.13}$$

Il s'agit en fait de contracter le signal en 0 afin de faire apparaître des périodes au cours des quelles la commande de vol n'est pas utilisée. On peut calculer la variance du signal suite à cette opération comme suit :

$$Var(c_M(x)) = Var(x)\left[(1+\epsilon^2)(1-\text{erf}(\frac{1}{\sqrt{2}}\epsilon)) - \epsilon\sqrt{\frac{2}{\pi}}e^{-\frac{1}{2}\epsilon^2}\right] \tag{A.14}$$
$$= Var(x)\left[1 - 2\sqrt{\frac{2}{\pi}}\epsilon + \epsilon^2 - \frac{1}{3}\sqrt{\frac{2}{\pi}}\epsilon^3 + \underset{\epsilon\to 0}{O}(\epsilon^5)\right] \tag{A.15}$$

où erf est la fonction d'erreur ($\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$) et ϵ est le rapport $\frac{\text{M}}{std(x)}$. L'amplitude est corrigée à posteriori pour conserver la valeur initiale.

A.2 Fonction de coût

À l'issue de chaque étape d'estimation, les filtres fournissent une valeur estimée de l'état X_n^+ ainsi que matrice de covariance P_n^+ qui décrit l'incertitude du résultat. Ces deux éléments décrivent en fait une distribution gaussienne dont la densité sur l'ensemble des états s'écrit :

$$f_{\text{X}_n^+,\text{P}_n^+}(\text{X}) = \frac{1}{(2\pi)^{\frac{\text{N}}{2}} |\text{P}_n^+|^{\frac{1}{2}}} e^{-\frac{1}{2}\|\text{X}_n^+ - \text{X}\|^2_{\text{P}_n^+}} \tag{A.16}$$

Où N est la dimension de l'espace des états, $|\cdot|$ est l'opérateur déterminant et $\|\cdot\|_\Sigma$ est la norme de Mahalanobis associée à la matrice Σ : $\|x\|^2_\Sigma = x^\text{T}\Sigma^{-1}x$.

En apprentissage, lorsque l'on veut attribuer une fonction de coût à un algorithme qui exprime des probabilités discrètes (comme en classification), il est courant d'utiliser l'entropie croisée entre la distribution attendue et la distribution exprimée. Pour des probabilités discrètes, la distribution attendue est généralement concentrée sur une seule valeur, et l'entropie vaut donc $-\log(\bar{p})$ où $\bar{p}$ est la probabilité que l'algorithme a exprimé pour la valeur attendue. Pour des probabilités continues le même principe peut s'appliquer en utilisant cette fois une distribution de Dirac centrée sur la valeur attendue. On retrouve alors une entropie qui s'exprime par $-\log(\bar{f})$ où $\bar{f}$ est la densité de probabilité que l'algorithme a exprimé pour la valeur attendue.

Dans le cas de notre distribution gaussienne, l'entropie prend donc cette forme :

$$\mathscr{H}_{\text{X}_n^+,\text{P}_n^+}(\text{X}) = \frac{1}{2}\|\text{X}_n^+ - \text{X}\|^2_{\text{P}_n^+} + \frac{1}{2}\log|\text{P}_n^+| + \frac{\text{N}}{2}\log(2\pi) \tag{A.17}$$

On retrouve la distance de Mahalanobis qui va pénaliser l'erreur commise par l'algorithme tout en le "pardonnant" lorsqu'il exprime une plus grande incertitude, le logarithme du déterminant de la matrice de covariance qui lui va pénaliser l'incertitude de l'algorithme et un terme constant que l'on supprimera puisqu'il n'a aucune conséquence sur l'apprentissage.

A.3 Définition formelle des modèles cinématiques

A.3.1 Modèle Rectiligne Uniforme (MRU)

$$\text{A} = \begin{pmatrix} 0_{3,3} & \text{I}_3 \\ 0_{3,3} & 0_{3,3} \end{pmatrix} \tag{A.18}$$

$$F(\Delta t) = \begin{pmatrix} I_3 & \Delta t I_3 \\ 0_{3,3} & I_3 \end{pmatrix} \tag{A.19}$$

$$Q = \begin{pmatrix} Q_P & C_{P,P'} \\ C_{P,P'}^T & Q_{P'} \end{pmatrix} \tag{A.20}$$

$$Q_{\Delta t} = \begin{pmatrix} \Delta t Q_P + \frac{1}{2}\Delta t^2 (C_{P,P'} + C_{P,P'}^T) + \frac{1}{3}\Delta t^3 Q_{P'} & \Delta t C_{P,P'} + \frac{1}{2}\Delta t^2 Q_{P'} \\ \Delta t C_{P,P'}^T + \frac{1}{2}\Delta t^2 Q_{P'} & \Delta t Q_{P'} \end{pmatrix} \tag{A.21}$$

A.3.2 Modèle Uniformément Accéléré (MUA)

$$A = \begin{pmatrix} 0_{3,3} & I_3 & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & I_3 \\ 0_{3,3} & 0_{3,3} & 0_{3,3} \end{pmatrix} \tag{A.22}$$

$$F(\Delta t) = \begin{pmatrix} I_3 & \Delta t I_3 & \frac{1}{2}\Delta t^2 I_3 \\ 0_{3,3} & I_3 & \Delta t I_3 \\ 0_{3,3} & 0_{3,3} & I_3 \end{pmatrix} \tag{A.23}$$

$$Q = \begin{pmatrix} Q_P & C_{P,P'} & C_{P,P''} \\ C_{P,P'}^T & Q_{P'} & C_{P',P''} \\ C_{P,P'}^T & C_{P,P''}^T & Q_{P''} \end{pmatrix} \tag{A.24}$$

$$Q_{\Delta t} = \begin{pmatrix} Q_{\Delta t,P} & C_{\Delta t,P,P'} & C_{\Delta t,P,P''} \\ C_{\Delta t,P,P'}^T & Q_{\Delta t,P'} & C_{\Delta t,P',P''} \\ C_{\Delta t,P,P'}^T & C_{\Delta t,P,P''}^T & Q_{\Delta t,P''} \end{pmatrix} \tag{A.25}$$

$$\begin{aligned} Q_{\Delta t,P} = {} & \Delta t Q_P + \frac{1}{3}\Delta t^3 Q_{P'} + \frac{1}{20} t^5 Q_{P''} + \frac{1}{2}\Delta t^2 (C_{P,P'} + C_{P,P'}^T) \\ & + \frac{1}{6}\Delta t^3 (C_{P,P''} + C_{P,P''}^T) + \frac{1}{8}\Delta t^4 (C_{P',P''} + C_{P',P''}^T) \end{aligned} \tag{A.26}$$

$$Q_{\Delta t,P'} = \Delta t Q_{P'} + \frac{1}{3}\Delta t^3 Q_{P''} + \frac{1}{2}\Delta t^2 (C_{P',P''} + C_{P',P''}^T) \tag{A.27}$$

$$Q_{\Delta t,P''} = \Delta t Q_{P''} \tag{A.28}$$

$$\begin{aligned} C_{\Delta t,P,P'} = {} & \Delta t C_{P,P'} + \frac{1}{2}\Delta t^2 Q_{P'} + \frac{1}{8}\Delta t^4 Q_{P''} \\ & + \frac{1}{2}\Delta t^2 C_{\Delta t,P,P''} + \frac{1}{6}\Delta t^3 (2C_{\Delta t,P',P''} + C_{\Delta t,P',P''}^T) \end{aligned} \tag{A.29}$$

$$C_{\Delta t,P,P''} = \Delta t C_{P,P''} + \frac{1}{6}\Delta t^2 Q_{P''} + \frac{1}{2}\Delta t^2 C_{P',P''} \tag{A.30}$$

$$C_{\Delta t,P',P''} = \Delta t C_{P',P''} + \frac{1}{2}\Delta t^2 Q_{P''} \tag{A.31}$$

A.3.3 Modèle de Singer

$$A = \begin{pmatrix} 0_{3,3} & I_3 & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & I_3 \\ 0_{3,3} & 0_{3,3} & -\alpha I_3 \end{pmatrix} \tag{A.32}$$

$$\begin{aligned} \mathscr{I}_0(t) &= e^{-\alpha t} \\ \mathscr{I}_1(t) &= \frac{1-e^{-\alpha\Delta t}}{\alpha} \\ \mathscr{I}_2(t) &= \frac{e^{-\alpha\Delta t}-1+\alpha\Delta t}{\alpha^2} \\ \mathscr{I}_3(t) &= \frac{1-\alpha\Delta t+\frac{1}{2}(\alpha\Delta t)^2-e^{-\alpha\Delta t}}{\alpha^3} \\ \mathscr{I}_4(t) &= \frac{e^{-\alpha\Delta t}-1+\alpha\Delta t-\frac{1}{2}(\alpha\Delta t)^2+\frac{1}{6}(\alpha\Delta t)^3}{\alpha^4} \end{aligned} \tag{A.33}$$

$$F(\Delta t) = \begin{pmatrix} I_3 & \Delta t I_3 & \mathscr{I}_2(t) I_3 \\ 0_{3,3} & I_3 & \mathscr{I}_1(t) I_3 \\ 0_{3,3} & 0_{3,3} & \mathscr{I}_0(t) I_3 \end{pmatrix} \tag{A.34}$$

$$Q = \begin{pmatrix} Q_P & C_{P,P'} & C_{P,P''} \\ C_{P,P'}^T & Q_{P'} & C_{P',P''} \\ C_{P,P''}^T & C_{P',P''}^T & Q_{P''} \end{pmatrix} \tag{A.35}$$

$$Q_{\Delta t} = \begin{pmatrix} Q_{\Delta t,P} & C_{\Delta t,P,P'} & C_{\Delta t,P,P''} \\ C_{\Delta t,P,P'}^T & Q_{\Delta t,P'} & C_{\Delta t,P',P''} \\ C_{\Delta t,P,P''}^T & C_{\Delta t,P',P''}^T & Q_{\Delta t,P''} \end{pmatrix} \tag{A.36}$$

$$\begin{aligned} Q_{\Delta t,P} =& \Delta t Q_P + \frac{1}{3}\Delta t^3 Q_{P'} + (\frac{2t}{\alpha}\mathscr{I}_3(t) - \frac{1}{2\alpha}\mathscr{I}_4(2t)) Q_{P''} \\ &+ \frac{1}{2}\Delta t^2 (C_{P,P'} + C_{P,P'}^T) + \mathscr{I}_3(t)(C_{P,P''} + C_{P,P''}^T) \\ &+ (t\mathscr{I}_3(t) - \mathscr{I}_4(t))(C_{P',P''} + C_{P',P''}^T) \end{aligned} \tag{A.37}$$

$$Q_{\Delta t,P'} = \Delta t Q_{P'} + (\frac{1}{2\alpha}\mathscr{I}_2(2t) - \frac{2}{\alpha}\mathscr{I}_2(t)) Q_{P''} + \mathscr{I}_2(t)(C_{P',P''} + C_{P',P''}^T) \tag{A.38}$$

$$Q_{\Delta t,P''} = \frac{1}{2}\mathscr{I}_1(2t) Q_{P''}$$

$$\begin{aligned} C_{\Delta t,P,P'} =& \Delta t C_{P,P'} + \mathscr{I}_2(t) C_{P,P''} + (t\mathscr{I}_2(t) - \mathscr{I}_3(t)) C_{P',P''} + \mathscr{I}_3(t) C_{P',P''}^T \\ &+ \frac{1}{2}\Delta t^2 Q_{\Delta t,P'} + \frac{1}{2}\mathscr{I}_2^2(t) Q_{P''} \end{aligned} \tag{A.39}$$

$$\begin{aligned} C_{\Delta t,P,P''} =& \mathscr{I}_1(t) C_{P,P''} + (t\mathscr{I}_1(t) - \mathscr{I}_2(t)) C_{P',P''} \\ &+ (\frac{1}{2\alpha^2}\mathscr{I}_1(2t) - \frac{t}{\alpha^2}\mathscr{I}_0(t)) Q_{P''} \end{aligned} \tag{A.40}$$

$$C_{\Delta t,P',P''} = \mathscr{I}_1(t) C_{P',P''} + \frac{1}{2}\mathscr{I}_1^2(t) Q_{P''} \tag{A.41}$$

A.3.4 Modèle MCU

$$A_\gamma = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\gamma & 0 \\ 0 & 0 & 0 & \gamma & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.42}$$

$$F(\Delta t) = \begin{pmatrix} 1 & 0 & 0 & \frac{1}{\gamma}\sin(\gamma\Delta t) & -\frac{1}{\gamma}(1-\cos(\gamma\Delta t)) & 0 \\ 0 & 1 & 0 & \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) & \frac{1}{\gamma}\sin(\gamma\Delta t) & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & \cos(\gamma\Delta t) & -\sin(\gamma\Delta t) & 0 \\ 0 & 0 & 0 & \sin(\gamma\Delta t) & \cos(\gamma\Delta t) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.43}$$

$$Q = \begin{pmatrix} q_X & c_{X,Y} & c_{X,Z} & c_{X,V_X} & c_{X,V_Y} & c_{X,V_Z} \\ c_{X,Y} & q_Y & c_{Y,Z} & c_{Y,V_X} & c_{Y,V_Y} & c_{Y,V_Z} \\ c_{X,Z} & c_{Y,Z} & q_Z & c_{Z,V_X} & c_{Z,V_Y} & c_{Z,V_Z} \\ c_{X,V_X} & c_{Y,V_X} & c_{Z,V_X} & q_{V_X} & c_{V_X,V_Y} & c_{V_X,V_Z} \\ c_{X,V_Y} & c_{Y,V_Y} & c_{Z,V_Y} & c_{V_X,V_Y} & q_{V_Y} & c_{V_Y,V_Z} \\ c_{X,V_Z} & c_{Y,V_Z} & c_{Z,V_Z} & c_{V_X,V_Z} & c_{V_Y,V_Z} & q_{V_Z} \end{pmatrix} \tag{A.44}$$

$$Q_{\Delta t} = \begin{pmatrix} q_{\Delta t,X} & c_{\Delta t,X,Y} & c_{\Delta t,X,Z} & c_{\Delta t,X,V_X} & c_{\Delta t,X,V_Y} & c_{\Delta t,X,V_Z} \\ c_{\Delta t,X,Y} & q_{\Delta t,Y} & c_{\Delta t,Y,Z} & c_{\Delta t,Y,V_X} & c_{\Delta t,Y,V_Y} & c_{\Delta t,Y,V_Z} \\ c_{\Delta t,X,Z} & c_{\Delta t,Y,Z} & q_{\Delta t,Z} & c_{\Delta t,Z,V_X} & c_{\Delta t,Z,V_Y} & c_{\Delta t,Z,V_Z} \\ c_{\Delta t,X,V_X} & c_{\Delta t,Y,V_X} & c_{\Delta t,Z,V_X} & q_{\Delta t,V_X} & c_{\Delta t,V_X,V_Y} & c_{\Delta t,V_X,V_Z} \\ c_{\Delta t,X,V_Y} & c_{\Delta t,Y,V_Y} & c_{\Delta t,Z,V_Y} & c_{\Delta t,V_X,V_Y} & q_{\Delta t,V_Y} & c_{\Delta t,V_Y,V_Z} \\ c_{\Delta t,X,V_Z} & c_{\Delta t,Y,V_Z} & c_{\Delta t,Z,V_Z} & c_{\Delta t,V_X,V_Z} & c_{\Delta t,V_Y,V_Z} & q_{\Delta t,V_Z} \end{pmatrix} \tag{A.45}$$

$$\begin{aligned} q_{\Delta t,\mathrm{X}} =& \Delta t\, q_{\mathrm{X}} + \frac{1}{2\gamma^3}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t))\, q_{\mathrm{V_X}} \\ &+ \frac{1}{2\gamma^3}(3\gamma\Delta t + \sin(\gamma\Delta t)(\cos(\gamma\Delta t) - 4))\, q_{\mathrm{V_Y}} \\ &+ \frac{2}{\gamma^2}(1 - \cos(\gamma\Delta t))\, c_{\mathrm{X,V_X}} - \frac{2}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t))\, c_{\mathrm{X,V_Y}} \\ &- \frac{1}{\gamma^3}(1 - \cos(\gamma\Delta t))^2\, c_{\mathrm{V_X,V_Y}} \end{aligned} \tag{A.46}$$

$$\begin{aligned} q_{\Delta t,\mathrm{Y}} =& \Delta t\, q_{\mathrm{Y}} + \frac{1}{2\gamma^3}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t))\, q_{\mathrm{V_Y}} \\ &+ \frac{1}{2\gamma^3}(3\gamma\Delta t + \sin(\gamma\Delta t)(\cos(\gamma\Delta t) - 4))\, q_{\mathrm{V_X}} \\ &+ \frac{2}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t))\, c_{\mathrm{Y,V_X}} + \frac{2}{\gamma^2}(1 - \cos(\gamma\Delta t))\, c_{\mathrm{X,V_Y}} \\ &+ \frac{1}{\gamma^3}(1 - \cos(\gamma\Delta t))^2\, c_{\mathrm{V_X,V_Y}} \end{aligned} \tag{A.47}$$

$$q_{\Delta t,\mathrm{Z}} = \Delta t\, q_{\mathrm{Z}} + \frac{1}{3}\Delta t^3\, q_{\mathrm{V_Z}} + \Delta t^2\, c_{\mathrm{Z,V_Z}} \tag{A.48}$$

$$\begin{aligned} c_{\Delta t,\mathrm{X,Y}} =& \frac{1}{2\gamma^3}(1 - \cos(\gamma\Delta t))^2 (q_{\mathrm{V_X}} - q_{\mathrm{V_Y}}) + \Delta t\, c_{\mathrm{X,Y}} \\ &+ \frac{1}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t))(c_{\mathrm{X,V_X}} - c_{\mathrm{Y,V_Y}}) \\ &+ \frac{1}{\gamma^2}(1 - \cos(\gamma\Delta t))(c_{\mathrm{X,V_Y}} + c_{\mathrm{Y,V_X}}) \\ &- \frac{1}{\gamma^3}(\gamma\Delta t - \sin(\gamma\Delta t)(2 - \cos(\gamma\Delta t)))\, c_{\mathrm{V_X,V_Y}} \end{aligned} \tag{A.49}$$

$$\begin{aligned} c_{\Delta t,\mathrm{X,Z}} =& \Delta t\, c_{\mathrm{X,Z}} + \frac{1}{2}\Delta t^2\, c_{\mathrm{X,V_Z}} + \frac{1}{\gamma^2}(1 - \cos(\gamma\Delta t))\, c_{\mathrm{Z,V_X}} \\ &- \frac{1}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t))\, c_{\mathrm{Z,V_Y}} \\ &+ \frac{1}{\gamma^3}(\sin(\gamma\Delta t) - \gamma\Delta t\cos(\gamma\Delta t))\, c_{\mathrm{V_X,V_Z}} \\ &- \frac{1}{\gamma^3}(\frac{1}{2}(\gamma\Delta t)^2 + 1 - \cos(\gamma\Delta t) - \gamma\Delta t\sin(\gamma\Delta t))\, c_{\mathrm{V_Y,V_Z}} \end{aligned} \tag{A.50}$$

$$\begin{aligned} c_{\Delta t,\mathrm{Y,Z}} =& \Delta t\, c_{\mathrm{Y,Z}} + \frac{1}{2}\Delta t^2\, c_{\mathrm{Y,V_Z}} + \frac{1}{\gamma^2}(1 - \cos(\gamma\Delta t))\, c_{\mathrm{Z,V_Y}} \\ &+ \frac{1}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t))\, c_{\mathrm{Z,V_X}} \\ &+ \frac{1}{\gamma^3}(\sin(\gamma\Delta t) - \gamma\Delta t\cos(\gamma\Delta t))\, c_{\mathrm{V_Y,V_Z}} \\ &+ \frac{1}{\gamma^3}(\frac{1}{2}(\gamma\Delta t)^2 + 1 - \cos(\gamma\Delta t) - \gamma\Delta t\sin(\gamma\Delta t))\, c_{\mathrm{V_X,V_Z}} \end{aligned} \tag{A.51}$$

$$\begin{aligned} c_{\Delta t,\mathrm{X},\mathrm{V_X}} =& \frac{1}{2\gamma^2}\sin(\gamma\Delta t)^2 q_{\mathrm{V_X}} + \frac{1}{2\gamma^2}(1-\cos(\gamma\Delta t))^2 q_{\mathrm{V_Y}} + \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{X},\mathrm{V_X}} \\ & - \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{X},\mathrm{V_Y}} + \frac{1}{\gamma^2}\sin(\gamma\Delta t)(1-\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Y}} \end{aligned} \tag{A.52}$$

$$\begin{aligned} c_{\Delta t,\mathrm{Y},\mathrm{V_Y}} =& \frac{1}{2\gamma^2}\sin(\gamma\Delta t)^2 q_{\mathrm{V_Y}} + \frac{1}{2\gamma^2}(1-\cos(\gamma\Delta t))^2 q_{\mathrm{V_X}} + \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{Y},\mathrm{V_Y}} \\ & + \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{Y},\mathrm{V_X}} + \frac{1}{\gamma^2}\sin(\gamma\Delta t)(1-\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Y}} \end{aligned} \tag{A.53}$$

$$\begin{aligned} c_{\Delta t,\mathrm{X},\mathrm{V_Y}} =& \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{X},\mathrm{V_Y}} + \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{X},\mathrm{V_X}} \\ & + \frac{1}{\gamma^2}\cos(\gamma\Delta t)(1-\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Y}} \\ & + \frac{1}{2\gamma^2}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t)) q_{\mathrm{V_X}} \\ & + \frac{1}{2\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t)(2-\cos(\gamma\Delta t))) q_{\mathrm{V_Y}} \end{aligned} \tag{A.54}$$

$$\begin{aligned} c_{\Delta t,\mathrm{Y},\mathrm{V_X}} =& \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{Y},\mathrm{V_X}} - \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{Y},\mathrm{V_Y}} \\ & + \frac{1}{\gamma^2}\cos(\gamma\Delta t)(1-\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Y}} \\ & - \frac{1}{2\gamma^2}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t)) q_{\mathrm{V_Y}} \\ & - \frac{1}{2\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t)(2-\cos(\gamma\Delta t))) q_{\mathrm{V_X}} \end{aligned} \tag{A.55}$$

$$c_{\Delta t,\mathrm{Z},\mathrm{V_Z}} = \Delta t\, c_{\mathrm{Z},\mathrm{V_Z}} + \frac{1}{2}\Delta t^2 q_{\mathrm{V_Z}} \tag{A.56}$$

$$c_{\Delta t,\mathrm{X},\mathrm{V_Z}} = \Delta t\, c_{\mathrm{X},\mathrm{V_Z}} + \frac{1}{\gamma^2}(1-\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Z}} - \frac{1}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t)) c_{\mathrm{V_Y},\mathrm{V_Z}} \tag{A.57}$$

$$c_{\Delta t,\mathrm{Y},\mathrm{V_Z}} = \Delta t\, c_{\mathrm{Y},\mathrm{V_Z}} + \frac{1}{\gamma^2}(1-\cos(\gamma\Delta t)) c_{\mathrm{V_Y},\mathrm{V_Z}} + \frac{1}{\gamma^2}(\gamma\Delta t - \sin(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Z}} \tag{A.58}$$

$$\begin{aligned} c_{\Delta t,\mathrm{Z},\mathrm{V_X}} =& \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{Z},\mathrm{V_X}} - \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{Z},\mathrm{V_Y}} \\ & + \frac{1}{\gamma^2}(\gamma\Delta t\sin(\gamma\Delta t) - (1-\cos(\gamma\Delta t))) c_{\mathrm{V_X},\mathrm{V_Z}} \\ & - \frac{1}{\gamma^2}(\sin(\gamma\Delta t) - \gamma\Delta t\cos(\gamma\Delta t)) c_{\mathrm{V_Y},\mathrm{V_Z}} \end{aligned} \tag{A.59}$$

$$\begin{aligned} c_{\Delta t,\mathrm{Z},\mathrm{V_Y}} =& \frac{1}{\gamma}\sin(\gamma\Delta t) c_{\mathrm{Z},\mathrm{V_Y}} + \frac{1}{\gamma}(1-\cos(\gamma\Delta t)) c_{\mathrm{Z},\mathrm{V_X}} \\ & + \frac{1}{\gamma^2}(\gamma\Delta t\sin(\gamma\Delta t) - (1-\cos(\gamma\Delta t))) c_{\mathrm{V_Y},\mathrm{V_Z}} \\ & + \frac{1}{\gamma^2}(\sin(\gamma\Delta t) - \gamma\Delta t\cos(\gamma\Delta t)) c_{\mathrm{V_X},\mathrm{V_Z}} \end{aligned} \tag{A.60}$$

$$q_{\Delta t,V_X} = \frac{1}{2\gamma}(\gamma\Delta t + \cos(\gamma\Delta t)\sin(\gamma\Delta t))q_{V_X} + \frac{1}{2\gamma}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t))q_{V_Y} - \frac{1}{\gamma}\sin(\gamma\Delta t)^2 c_{V_X,V_Y} \tag{A.61}$$

$$q_{\Delta t,V_Y} = \frac{1}{2\gamma}(\gamma\Delta t - \cos(\gamma\Delta t)\sin(\gamma\Delta t))q_{V_X} + \frac{1}{2\gamma}(\gamma\Delta t + \cos(\gamma\Delta t)\sin(\gamma\Delta t))q_{V_Y} + \frac{1}{\gamma}\sin(\gamma\Delta t)^2 c_{V_X,V_Y} \tag{A.62}$$

$$q_{\Delta t,V_Z} = \Delta t\, q_{V_Z} \tag{A.63}$$

$$c_{\Delta t,V_X,V_Y} = \frac{1}{\gamma}\cos(\gamma\Delta t)\sin(\gamma\Delta t)c_{V_X,V_Y} + \frac{1}{2\gamma}\sin(\gamma\Delta t)^2(q_{V_X} - q_{V_Y}) \tag{A.64}$$

$$c_{\Delta t,V_X,V_Z} = \frac{1}{\gamma}\sin(\gamma\Delta t)c_{V_X,V_Z} - \frac{1}{\gamma}(1-\cos(\gamma\Delta t))c_{V_Y,V_Z} \tag{A.65}$$

$$c_{\Delta t,V_Y,V_Z} = \frac{1}{\gamma}\sin(\gamma\Delta t)c_{V_Y,V_Z} + \frac{1}{\gamma}(1-\cos(\gamma\Delta t))c_{V_X,V_Z} \tag{A.66}$$

RÉSUMÉ

Cette thèse cherche à explorer les possibilités offertes par l'utilisation de méthode d'apprentissage automatique dans le domaine du pistage radar. L'algorithme le plus largement utilisé pour estimer en temps réel l'état de la cible suivie est le filtre de Kalman ou l'une de ses variations (EKF, IMM...). Son bon fonctionnement repose toutefois sur une connaissance accrue du modèle cinématique d'évolution de la cible. Ce dernier constat soulève deux problèmes, tout d'abord la connaissance du modèle ne peut qu'être lacunaire tant que le type de cible n'a pas été identifié, ensuite il semble laborieux et peu économe d'établir manuellement un modèle adapté pour chaque type de cible susceptible d'être rencontré. Ce sont ces difficultés que la thèse cherche à pallier par l'utilisation de réseaux de neurones.

La première contribution est l'utilisation de réseaux de neurones profonds pour des tâches de classification des types d'aéronefs basées sur des trajectoires réelles issues des transmissions ADS-B. La seconde est la création d'un jeu de données simulées permettant l'entraînement des algorithmes de pistage prenant en entrée des mesures de positions bruitées dont l'incertitude est connue et cherchant à estimer l'état de la cible (position et vitesse). La dernière et principale est l'implémentation, l'entraînement et l'analyse des différents filtres, qu'ils soient issus de la littérature ou proposés ici. Deux familles de filtres ont ainsi été étudiées, les filtres adaptatifs dont les paramètres évoluent dynamiquement en fonction des besoins, et les filtres à modèles multiples qui mettent en émulation plusieurs filtres simples.

MOTS CLÉS

Apprentissage automatique, filtres de Kalman, réseaux de neurones, classification

ABSTRACT

This thesis aims to explore the possibilities offered by the use of machine learning in the field of radar tracking. The most widely used algorithm when trying to estimate the state of the target in real time is the Kalman filter or one of its variations (EKF, IMM...). However, its effective functioning relies on a greater insight into the kinematic model of the target's state evolution. This latter observation leads to two main issues, first the knowledge of the kinematic model can only be deficient as long as the taget has not been classified, second it looks arduous and greedy to manually set a fitted model for each type of aircraft one is prone to encounter. These are the two difficulties this thesis aims to compensate for by using neural networks.

The first contribution is the use of deep neural networks for aircrafts classification tasks based on real trajectories that stem from ADS-B transmissions. The second contribution is the creation of a simulated dataset allowing for the training of tracking algorithms taking as input noisy position measurements with known uncertainty and aiming to estimate the state of the target (position and speed). Le last and main contribution is the implementation, training and analysis of the various filters whether they are taken from the literature or proposed in this thesis. Two families of filters have been studied this way, the adaptive filters which are able to dynamically adjust their parameters depending on the situation, and the multiple models filters which make different simple filters run and interact with each others.

KEYWORDS

Machine learning, Kalman filters, neural networks, classification