Université Paris-Sud 11
Laboratoire des Signaux et Systèmes (L2S)

Thèse

présentée en première version en vue d'obtenir le grade de
Docteur, spécialité « Physique »

par

Thomas Boulay

Développement d'algorithmes pour la fonction NCTR Application des calculs parallèles sur les processeurs GPU

Thèse soutenue le 29 Octobre 2013 devant le jury composé de :

M.	Christian Brousseau	Université de Rennes 1	(Rapporteur)
M.	Dominique Houzet	Grenoble INP	(Rapporteur)
M.	Ali Mohammad-Djafari	L2S	(Directeur)
M.	Nicolas Gac	L2S	(Examinateur)
M.	Julien Lagoutte	Thales Air Systems	(Examinateur)
M.	François Le Chevalier	Thales Air Operations	(Examinateur)

Remerciements

Je commence par remercier avant tout Thales Air Systems, mon laboratoire d'accueil, le L2S, ainsi que l'ANRT pour m'avoir offert l'opportunité d'effectuer cette thèse CIFRE. Je remercie très sincérement mon directeur de thèse, Ali Mohammad-Djafari pour l'aide qu'il m'a apportée. Son expérience et sa pédagogie m'ont permis d'organiser au mieux mon travail et ont été pour moi sources de motivation et de curiosité durant ces trois années. Je remercie également tout particulièrement mon responsable à Thales Air Systems, Julien Lagoutte, pour l'implication et la générosité qu'il a pris soin d'apporter au suivi de mes travaux de thèse. Ses compétences techniques, sa rigueur et son dynamisme m'ont permis d'avancer plus efficacement sur mon sujet de thèse. Je tiens également à remercier Nicolas Gac, maître de conférence et membre du GPI, pour son soutien et sa disponibilité tout au long de la thèse. Ses compétences dans le domaine des GPUs m'ont été très utiles pour aborder cette spécialité qui avant le début de la thèse ne m'était pas du tout familière. Son expérience et son implication m'ont été également très précieuses pour organiser mes travaux de thèse.

Je remercie aussi les membres du département Etudes Amont de Thales Air Systems et toutes les autres personnes que j'ai pu cotoyer au cours des périodes passées sur le site de Limours, qui par leur accueil et leur gentillesse ont rendu les journées de travail beaucoup plus agréables. Je remercie en particulier Michel Moruzzis et François Giersch qui ont bien voulu consacrer une partie de leur précieux temps pour apporter leur expertise à mes travaux de thèse.

Je tiens également à remercier tous les thésards ou post-doc que j'ai croisés au L2S (dans l'ordre chronologique Doriano, Ning, Jerôme, Jean-François, Caifang, Yuling, Sha, Remi, Long, Leila) ainsi que tous les stagiaires. Je remercie également les membres du GPI qui par leur accueil et leur sympathie m'ont permis d'effectuer cette thèse dans les meilleures conditions.

Enfin, je remercie mes parents, mes frères et tous mes proches pour leur soutien indéfectible. Sans eux, il m'aurait été difficile d'aller jusqu'au bout.

Gif-sur-Yvette, le 23 août 2013.

Table des matières

Table des matières iv

Liste des figures viii

Liste des tableaux xi

Notations xv

Préface 1

1 Etat de l'art 5
- 1.1 Fonction NCTR . . . 6
  - 1.1.1 Inverse Synthetic Aperture Radar (ISAR) . . . 7
  - 1.1.2 Jet Engine Modulation (JEM) . . . 7
  - 1.1.3 Définition du taux d'erreur, taux de succès et taux de bonne identification . . . 8
- 1.2 Haute Résolution Distance (HRD) . . . 9
  - 1.2.1 Formes d'onde haute résolution . . . 10
  - 1.2.2 Obtention d'un profil distance HRD . . . 12
  - 1.2.3 Diffuseurs ou points brillants . . . 13
  - 1.2.4 Influence de l'angle d'aspect . . . 14
  - 1.2.5 Alignement des profils distance HRD . . . 17
- 1.3 Méthodes de classification . . . 21
  - 1.3.1 Classification non paramétrique . . . 22
  - 1.3.2 Classification paramétrique . . . 27
- 1.4 Espace de représentation . . . 35
  - 1.4.1 Profils distance haute résolution dans le domaine fréquentiel . . . 35
  - 1.4.2 Ondelettes . . . 37
  - 1.4.3 Représentations parcimonieuses . . . 38
- 1.5 Conclusion du chapitre . . . 39

2 Présentation des données et caractérisation statistique 41
- 2.1 Base de données . . . 41
  - 2.1.1 Données synthétiques . . . 41
  - 2.1.2 Données réelles . . . 43
- 2.2 Visualisation simple de la base de données synthétiques 43
- 2.3 Etude statistique des matrices de covariance . . . 46
  - 2.3.1 Inertie des matrices de covariance . . . 47
  - 2.3.2 Volume des classes . . . 50

2.3.3 Représentation graphique des données 53
2.4 Conclusion du chapitre 59

3 Algorithme des KPPV 61
3.1 Présentation générale 62
3.2 Première étape : calcul des métriques 62
3.3 Deuxième et troisième étapes : tri des métriques calculées à l'étape 1 et décision 63
3.4 Paramètres de réglage 64
3.5 Performances de l'algorithme des KPPV 64
3.5.1 $RSB = 30dB$ 65
3.5.2 $RSB = 20dB$ 65
3.5.3 $RSB = 15dB$ 66
3.5.4 Matrices d'acceptation (MA) ou matrice de confusion d'acceptation 66
3.6 Conclusion du chapitre 67

4 Algorithmes de reconnaissance de cibles basés sur les méthodes probabilistes 69
4.1 Modèle de mélange pour les applications NCTR 70
4.1.1 Gaussian Mixture Models (GMM) 70
4.1.2 Modèle de mélange de Student 71
4.2 Estimation des paramètres et calcul des probabilités *a posteriori* 72
4.2.1 Estimation des paramètres 72
4.2.2 Calcul des probabilités *a posteriori* 73
4.3 Règles de décision 74
4.3.1 Règle de décision du MAP 74
4.3.2 Règle de décision pour le contrôle du taux d'erreur . . . 75
4.4 Paramètres de réglages 75
4.5 Performances de l'algorithme basé sur les méthodes probabilistes 76
4.6 Représentation graphique des données via les méthodes probabilistes 78
4.6.1 Stochastic Neighbor Embedding (SNE) 78
4.6.2 Carte de probabilité dans l'espace de dimension réduite . 80
4.7 Conclusion du chapitre 81

5 Algorithmes de reconnaissance de cibles basés sur la logique floue 83
5.1 Distance Floue pour la Reconnaissance Instantanée (D-FRI) 84
5.1.1 Construction des fonctions d'appartenance 84
5.1.2 Reconnaissance du profil distance sous test 85
5.1.3 Performances 89
5.2 Gabarits flous pour la Reconnaissance Instantanée (G-FRI) 91
5.2.1 Etape 1 : construction de la base d'apprentissage 92
5.2.2 Etape 2 : mise en forme du profil distance sous test . . . 96
5.2.3 Etape 3 : comparaison du profil distance sous test avec la base d'apprentissage 97

5.2.4 Adaptation des gabarits aux autres classes de la base de données . . . 101
5.2.5 Performances . . . 101
5.3 Conclusion du chapitre . . . 103

6 Parallélisation sur GPU 105
6.1 Les processeurs graphiques (GPU) . . . 106
6.1.1 Historique : vers une architecture many-cores . . . 106
6.1.2 Puissance de calcul des GPUs . . . 107
6.1.3 Modèle de programmation sur GPUs . . . 110
6.1.4 Accès mémoire . . . 111
6.1.5 Génération Tesla vs génération Fermi . . . 112
6.2 Etat de l'art de l'utilisation des GPUs dans les domaines en lien avec la thèse . . . 113
6.2.1 Calcul de la SER sur GPU . . . 113
6.2.2 Reconnaissance de cibles sur GPU . . . 113
6.2.3 K plus proches voisins sur GPU . . . 114
6.2.4 Méthodes probabilistes sur GPU . . . 115
6.2.5 Logique floue sur GPU . . . 116
6.3 Algorithme des KPPV et parallélisation sur GPU . . . 117
6.3.1 Nombre d'opérations à virgule flottante . . . 118
6.3.2 Temps de calcul sous Matlab . . . 118
6.3.3 Découpage en thread sur l'architecture many-core GPU . 119
6.3.4 Latence, Occupation et parallélisme . . . 119
6.3.5 Implémentation CUDA C++ sur GPU . . . 123
6.3.6 Implémentation OpenCL . . . 129
6.3.7 Implémentation Matlab mex-CUDA et Matlab mex-OpenCL 129
6.3.8 Conclusion . . . 130
6.4 Méthodes probabilistes et parallélisation sur GPU . . 131
6.5 Algorithme logique floue et parallélisation sur GPU 134
6.5.1 Algorithme D-FRI . . . 134
6.5.2 Algorithme G-FRI . . . 135
6.5.3 Accélération de l'étape de calcul des possibilités . . . 137
6.6 Conclusion du chapitre . . . 138

7 Conclusion générale 139
7.1 Synthèse des propriétés des différents algorithmes présentés dans le mémoire . . . 139
7.2 Synthèse des performances des différents algorithmes présentés dans le mémoire . . . 144
7.3 Synthèse des performances de parallélisation . . . 145
7.4 Synthèse globale . . . 146
7.5 Perspectives . . . 147

A Annexes 149
A.1 Estimation des coûts de calcul . . . 150
A.1.1 Méthodes probabilistes . . . 150
A.1.2 Algorithme D-FRI . . . 152
A.1.3 Algorithme G-FRI . . . 153
A.2 Fonctions de calcul de la possibilité . . . 154

A.2.1 Pseudo-code de la première version de la fonction de calcul de la possibilité 154
A.2.2 Pseudo-code de la deuxième version de la fonction de calcul de la possibilité 156
A.2.3 Pseudo-code de la troisième version de la fonction de calcul de la possibilité 157
A.3 Publications liées à la thèse 160

Bibliographie **163**

Liste des figures

1 Principe des traitements NCTR 2

1.1 Règle de décision à trois niveaux 9
1.2 Un exemple de profil distance 9
1.3 Forme d'onde "Chirp" 11
1.4 Forme d'onde "Stepped-frequency" 11
1.5 Forme d'onde stretch 12
1.6 Formation du profil distance 13
1.7 Un exemple de diffuseurs sur une cible 14
1.8 Influence de l'angle d'aspect 15
1.9 Calcul de la possibilité (P) et de la nécessité (N) 34
1.10 Synthèse de l'état de l'art 40

2.1 Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 30dB$ 43
2.2 Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 20dB$ 44
2.3 Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 15dB$ 44
2.4 Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 30dB$ 45
2.5 Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 20dB$ 45
2.6 Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 15dB$ 46
2.7 Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 30dB$ 48
2.8 Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 20dB$ 49

2.9 Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 15dB$ 50
2.10 Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 30dB$ 55
2.11 Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 20dB$ 57
2.12 Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 15dB$ 58

3.1 Principe de l'algorithme des KPPV 62
3.2 Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 30dB$) 65
3.3 Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 20dB$) 65
3.4 Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 15dB$) 66

4.1 Visualisation 2D des données à partir de la méthode t-SNE pour les classes 1, 2 and 3 (lignes) et pour les trois niveaux de RSB (colonnes). 81

5.1 Construction des fonctions d'appartenance 85
5.2 Calcul des distances 87
5.3 Calcul des densités de possibilité 88
5.4 Calcul des possibilités 88
5.5 Positions moteur et gabarit 93
5.6 Positions moteur et profils distance 94
5.7 Fonction d'appartenance 95
5.8 Construction des données d'apprentissage 96
5.9 Densité de possibilité 97
5.10 Sélection des fonctions d'appartenance 98
5.11 Prise de décision finale 100

6.1 Puissance de calcul GPU (NVI 2012) 108
6.2 Débit mémoire sur GPU (NVI 2012) 109
6.3 Architecture CPU et GPU (NVI 2012) 109
6.4 Architecture mémoire des GPUs (NVI 2012) 111
6.5 Architecture mémoire des GPUs Fermi (NVI 2012) 112
6.6 Pseudo-Code du "kernel" 118
6.7 Utilisation des données 120
6.8 Parallélisme de threads (inspiré de (Volkov 2010)) 121
6.9 Parallélisme d'instructions (inspiré de (Volkov 2010)) 123

6.10 Alignement et séquentialité des données en mémoire (NVI 2012) 124
6.11 Stockage en mémoire de la base d'apprentissage 125
6.12 Pseudo-Code du "kernel" modifié pour un profil distance de test k et un profil distance d'apprentissage j 127
6.13 Pseudo-Code de l'algorithme probabiliste 132
6.14 Pseudo-Code de l'algorithme D-FRI 134
6.15 Pseudo-Code de l'algorithme G-FRI 136

7.1 Histogrammes des profils distance de la base d'apprentissage pour les trois classes. 142

A.1 Fonction d'appartenance et densité de possibilité 154
A.2 Pseudo-Code de la première version de la fonction de calcul de la possibilité 155
A.3 Pseudo-Code de la deuxième version de la fonction de calcul de la possibilité 156
A.4 Troisième version de la fonction de calcul de la possibilité . 158

Liste des tableaux

1.1 Différentes techniques NCTR 6

2.1 Notations utilisées pour la base d'apprentissage 47
2.2 Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 30dB$ 51
2.3 Caractérisation statistique des profils distance de la base de test pour un $RSB = 30dB$ 51
2.4 Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 20dB$ 51
2.5 Caractérisation statistique des profils distance de la base de test pour un $RSB = 20dB$ 52
2.6 Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 15dB$ 52
2.7 Caractérisation statistique des profils distance de la base de test pour un $RSB = 15dB$ 52
2.8 Distance euclidienne entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 30dB$ 54
2.9 Distance euclidienne entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 30dB$ 54
2.10 Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 20dB$ 56
2.11 Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 20dB$ 56
2.12 Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 15dB$ 57
2.13 Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 15dB$ 58

3.1 Matrice d'acceptation pour les trois niveaux de RSB - Algorithme des KPPV 67
3.2 Taux d'erreur et de succès - Algorithme des KPPV 67

4.1 Matrices de décision **MA**, **MI** and **MR** pour trois RSB différents dans l'espace de grande dimension - Algorithme bayésien 77
4.2 Taux d'erreur, de bonne identification et de succès - Algorithme bayésien 77

5.1 Matrices de décision **MA**, **MI** et **MR** pour trois RSB différents - Algorithme D-FRI 90
5.2 Taux d'erreur, de bonne identification et de succès - Algorithme D-FRI 90
5.3 Matrices de décision **MA**, **MI** et **MR** pour les données réelles - Algorithme D-FRI 91
5.4 Taux d'erreur, taux de succès et taux de bonne identification - Algorithme D-FRI 91
5.5 Matrices de décision **MA**, **MI** et **MR** pour les données réelles et uniquement pour la classe 1 - Algorithme G-FRI . 102
5.6 Taux d'erreur - Algorithme G-FRI 102
5.7 Matrices de décision **MA**, **MI** et **MR** pour les données réelles - Algorithme G-FRI 102
5.8 Taux d'erreur, taux de succès et taux de bonne identification - Algorithme G-FRI 103

6.1 Temps de calcul (en ms) de l'étape de calcul des distance, recalage et seuillage de l'algorithme des KPPV exécuté en simple precision sous Matlab pour 150 profils de test 119
6.2 Facteurs limitant l'occupation avec les cartes Fermi et Tesla 122
6.3 Calcul de l'occupation en fonction des facteurs limitant pour différentes valeurs de nombre de threads par bloc sur une carte Fermi 122
6.4 Propriétés et performances des différentes versions de l'algorithme des KPPV en simple precision sur GPU Fermi et Tesla en CUDA C++ 128
6.5 Caractéristiques des différentes versions sur la carte C2050 . 129
6.6 Etape de calcul des distances, recalage et seuillage de l'algorithme des KPPV en simple precision sur GPU Fermi et Tesla en OpenCL 129
6.7 Temps de calcul (en ms) de l'étape de calcul des distances, recalage et seuillage de l'algorithme des KPPV exécuté en simple precision sous Matlab, Matlab CUDA et Matlab OpenCL 130
6.8 Coût de calcul pour les trois étapes de l'algorithme probabiliste 132
6.9 FLOP pour chaque étape de l'algorithme basé sur l'approche probabiliste 132
6.10 Temps d'exécution total estimé pour l'algorithme basé sur les méthodes probabilistes ($K = 3$) 133
6.11 Temps d'exécution total estimé pour l'algorithme basé sur la logique floue ($K = 3$) 135
6.12 Temps d'exécution total estimé pour l'algorithme G-FRI ($K = 3$) 136
6.13 Temps de calcul (en ms) de 200 possibilités 137

7.1 Propriétés des trois approches étudiées pour nos algorithmes NCTR 140
7.2 Résultats de nos trois algorithmes NCTR pour une cible dont la classe n'appartient pas à la base de données 143

7.3 Résultats de nos trois algorithmes NCTR pour un profil distance de la classe 1 fortement bruité 143
7.4 Taux d'erreur, de bonne identification et de succès obtenues avec les trois différents types d'algorithmes étudiés durant la thèse et sur les données synthétiques 145
7.5 Taux d'erreur, taux de succès et taux de bonne identification pour les données réelles et pour les deux algorithmes basés sur la logique floue. 145

A.1 Valeurs des variables pour l'estimation des coûts de calcul. 150

Abréviations

ACI	Analyse en Composantes Indépendantes
ACP	Analyse en Composantes Principales
ACPP	Analyse en Composantes Principales Probabiliste
CPU	Conception Assistée par Ordinateur
CPU	Central Processing Unit
CUDA	Common Unified Device Architecture
D-FRI	Distance floue pour la Reconnaissance Instantanée
EM	Espérance-Maximisation
FA	Factor Analysis
FFT	Fast Fourier Transform
FLOP	FLoating point OPerations
FLOPS	FLoating Point Operations Per Second
G-FRI	Gabarits Flous pour la Reconnaissance Instantanée
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HERM	Helicopter Rotor Modulation
HRD	Haute Résolution Distance
IFF	Identification Friend or Foe
ILP	Instruction-Level Parallelism
IQ	In-phase and Quadrature
ISAR	Inverse Synthetic Aperture Radar
JEM	Jet Engine Modulation
KNN	K Nearest Neighbors
KPPV	K Plus Proches Voisins
LDA	Linear Discriminant Analysis
LFA	Local Factor Analysis
LLE	Locally Linear Embedding
LS	Lecture en SDRAM
MA	Matrices d'acceptation
MAP	Maximum A Posteriori
MCC	Maximum Correlation Coefficient
MCE	Minimum Classification Error
MFA	Mixture of Factor Analysis
MI	Matrices d'incertitude
MR	Matrices de rejet
MTRC	Migration Through Resolution Cells
MV	Maximum de Vraisemblance

MVU	Maximum Variance Unfolding
NCTR	Non Cooperative Target Recognition
NLR	Nombre de loads réels
NLT	Nombre de loads théoriques
OPENCL	Open Computing Language
PCT	Parallel Computing Toolbox
PT	Power Transformed
PTC	Power Transformed Correlation
RD	Réutilisation des données
RPCL	Rival Penalized Competitive Learning
RSB	Rapport Signal sur Bruit
RX	Module de réception
SA	Seuil d'acceptation
SAR	Synthetic Aperture Radar
SER	Surface Equivalente Radar
SR	Seuil de rejet
SIMT	Single Instruction Multiple Threads
SM	Streaming Multiprocessor
SNE	Stochastic Neighbor Embedding
SVM	Support Vector Machine
SZPR	Smoothed Zero Phase Representation
TBI	Taux de bonne identification
TE	Taux d'erreur
TLP	Thread-Level Parallelism
TS	Taux de succès
TX	Module de transmission

Introduction

Au cours des derniers conflits, les tirs fraticides ont causé d'importantes pertes humaines et la destruction de matériels (chasseurs, avions de ligne, etc). De plus, les erreurs d'identification sont toujours un risque potentiel de pertes humaines civiles. Avoir la bonne identification d'une cible est donc indispensable pour éviter ces pertes.

Aujourd'hui, deux types d'exploitation des données radars fonctionnellement très différents existent : les applications coopératives (IFF pour Identification Friend or Foe) et les applications non coopératives (NCTR pour Non Cooperative Target Recognition). Contrairement aux applications coopératives, les applications non coopératives sont réalisées sans collaboration de la cible. Deux niveaux de classification peuvent être définis. Le premier niveau que l'on appelle "classification" est un niveau où l'on souhaite distinguer des grandes classes de cibles (Hélicoptères, Avions de ligne, Chasseurs, etc...). Le deuxième niveau de classification est celui que l'on appelle plus communément "reconnaissance". Dans le cadre de la thèse, nous nous intéressons à ce second niveau de classification. La reconnaissance est plus fine que la classification. Il s'agit en effet de distinguer à l'intérieur d'une grande classe de cibles, le modèle de la cible. Par exemple dans la classe des chasseurs, on cherchera à distinguer un Rafale, d'un Mirage 2000 ou d'un F-16. Dans le cas non coopératif, ces deux niveaux sont regroupés sous le terme fonction NCTR. Les applications NCTR utilisent principalement des radars de type sol-air ou air-air pour faire la reconnaissance. La figure 1 illustre le principe général des traitements associés aux algorithmes NCTR.

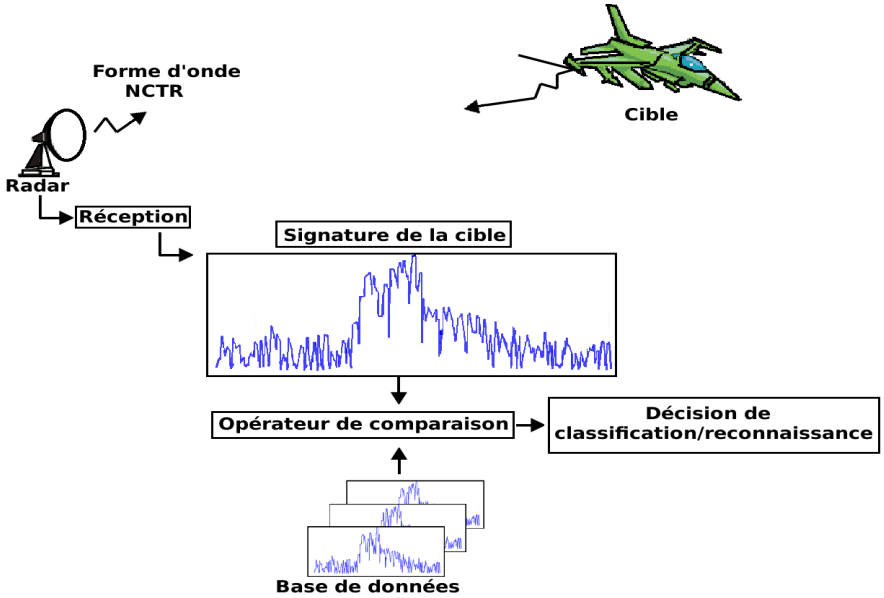


FIG. 1 : Principe des traitements NCTR

Les applications de reconnaissance non coopérative deviennent aujourd'hui indispensables au regard des possibles défaillances des applications de reconnaissance coopérative. En effet, ces dernières partent du principe que la coopération est toujours garantie. Or cela n'est pas toujours le cas en raison notamment, de problèmes techniques, d'erreurs humaines ou le plus souvent, en cas de conflits, d'intentions hostiles de la part de la cible à identifier.

Un des enjeux majeurs d'une fonction NCTR est de fournir le plus souvent une décision unique garantissant le taux d'erreur [1]. En effet, il n'est pas acceptable de ne pas garantir le taux d'erreur devant la gravité des conséquences d'une erreur de reconnaissance (pertes humaines, de civils, etc...). Ainsi, l'objectif d'un traitement NCTR consiste à maximiser le taux de succès (ou mieux le taux de bonne identification) tout en ne dépassant pas un taux d'erreur fixé par les considérations opérationnelles. Pour être capable de maîtriser au mieux le taux d'erreur tout en maximisant le taux de succès, des traitements adaptés doivent donc être mis en oeuvre. Ces traitements doivent par exemple tenir compte de la variation de l'amplitude du signal radar pour chaque point de la signature de la cible, ce qui est généralement dû à des erreurs de mesure mais aussi dans certains cas, par exemple, à la rotation des moteurs. La prise en compte de tous ces phénomènes entraîne des puissances de calcul élevées.

En parallèle, l'avènement des nouvelles technologies GPU, et leurs applications au calcul scientifique et temps réel, permet de reconsidérer les contraintes historiques qui pèsent sur le développement de la fonction NCTR dans les radars. Ces contraintes concernent notamment l'accès aux puissances de calcul nécessaires pour réaliser en temps réel les calculs nécessaires à la reconnaissance de la cible. Les solutions actuelles pour la fonction NCTR sont dans l'obligation d'avoir recours à des simplifications

[1] Les termes "erreur", "succès" et "bonne identification" sont définis dans la section 1.1.3.

(donc avec des performances limitées) pour les traitements NCTR temps réel. Pour reconnaître une cible, la signature radar de la cible est comparée avec des signatures existantes dans une base d'apprentissage. Du fait, des puissances de calculs induites, il y a une obligation de compresser l'information en réduisant la précision sur les signatures radars mesurées ou en réduisant le nombre de signatures utilisées dans la base d'apprentissage.

L'intérêt de la thèse réside dans la définition, la mise au point et l'évaluation de nouveaux traitements qui se rapprocheraient de l'algorithme optimal. Les technologies GPU rendent possible l'accès à des traitements optimaux (sans compression des données, c'est à dire de type "force brute") au sens des performances pour les applications temps réel. L'architecture des processeurs graphiques (GPU) de nouvelles générations présente alors une architecture particulièrement bien adaptée à ce type d'algorithme massivement parallèle. Il est important de noter que ces nouveaux algorithmes d'identification doivent être définis en tenant compte des contraintes suivantes :

- Maîtriser le taux d'erreur en-dessous d'un seuil fixé par l'opérateur ;
- Maximiser le taux de succès.

Nos contributions durant la thèse portent donc sur deux axes majeurs :

- Le premier est la définition d'algorithmes de reconnaissance de cibles non coopératives respectant les contraintes énoncées ci-dessus ;
- Le deuxième est l'évaluation des gains que peuvent apporter la parallélisation sur GPU d'algorithmes de reconnaissance de cibles non coopératives.

Le *chapitre 1* fait l'état de l'art des solutions existantes pour la reconnaissance de cibles non coopératives. Le *chapitre 2* revient sur les propriétés de la base de données utilisée. Le *chapitre 3* détaille un premier algorithme de reconnaissance de cibles non coopératives basé sur l'algorithme des KPPV. Dans le *chapitre 4*, nous nous intéressons aux méthodes probabilistes et à leurs applications dans le contexte de la reconnaissance de cibles non coopératives. Le *chapitre 5* présente deux algorithmes de reconnaissance de cibles non coopératives basés sur la logique floue. Nous abordons dans le *chapitre 6* les aspects parallélisation sur GPU. Nous terminons en présentant les conclusions et les perspectives qui ressortent du travail de thèse présenté.

Cette thèse a fait l'objet de divers travaux écrits : (i) un premier article intitulée *Algorithmes de Reconnaissance de Cibles Non coopératives et implémentation sur GPU* a été présenté lors de la conférence GRETSI 2011 à Bordeaux. (ii) Un article a également été présenté lors de la conférence SITIS (Signal Image Technologie and Internet based Systems) en 2012 à Naples. Cet article s'intitule *A Fuzzy-Logic based non cooperative target recognition*. (iii) Un article a été publié dans la revue Traitement du Signal en 2013. Le titre de cet article est "Algorithmes de reconnaissance NCTR et parallélisation sur GPU". (iv) Pour finir, un dernier article est en cours de rédaction pour la revue IET Radar, Sonar and Navigation. On retrouve une liste des articles publiés ou des présentations orales effectuées durant

la thèse en annexe (cf. section A.3).

Cette thèse a bénéficié d'un financement CIFRE de la part de l'Association Nationale Recherche Technologie (ANRT) et a été réalisée en collaboration avec Thales Air Systems et le Laboratoire des Signaux et Systèmes (L2S). Au sein de Thales Air Systems, l'environnement de travail a été celui du département Etudes Amont qui a pour objectifs d'étudier et valider de nouveaux concepts pour les futurs radars. Le cadre de travail au sein du L2S s'est lui concentré autour du Groupe Problèmes Inverses (GPI). Les activités du GPI concernent essentiellement la restauration et la reconstruction d'images ou de signaux (déconvolution, séparation de sources, etc...). Les applications s'étendent sur des domaines comme l'imagerie médicale ou satellitaire, l'astrophysique, la géophysique ou encore l'imagerie en contrôle non-destructif.

1 Etat de l'art : La classification des profils distance HRD au sein de la fonction NCTR

Sommaire

1.1 Fonction NCTR 6
1.1.1 Inverse Synthetic Aperture Radar (ISAR) 7
1.1.2 Jet Engine Modulation (JEM) 7
1.1.3 Définition du taux d'erreur, taux de succès et taux de bonne identification 8
1.2 Haute Résolution Distance (HRD) 9
1.2.1 Formes d'onde haute résolution 10
1.2.2 Obtention d'un profil distance HRD 12
1.2.3 Diffuseurs ou points brillants 13
1.2.4 Influence de l'angle d'aspect 14
1.2.5 Alignement des profils distance HRD 17
1.3 Méthodes de classification 21
1.3.1 Classification non paramétrique 22
1.3.2 Classification paramétrique 27
1.4 Espace de représentation 35
1.4.1 Profils distance haute résolution dans le domaine fréquentiel 35
1.4.2 Ondelettes 37
1.4.3 Représentations parcimonieuses 38
1.5 Conclusion du chapitre 39

Le but de ce chapitre est de faire l'état de l'art des solutions existantes dans le domaine de la reconnaissance de cibles non coopératives (NCTR). Dans la plupart des algorithmes de classification et en particulier dans les algorithmes de reconnaissance de cibles non coopératives, il existe deux problématiques principales : (i) la sélection des attributs pour faire la classification et (ii) le classifieur. Généralement, les paramètres d'entrée d'un classifieur sont les attributs extraits lors d'une première étape auxquels

on peut ajouter de la connaissance *a priori*. En sortie, le classifieur fournit l'identité de la cible à laquelle, selon les types de classifieur, on associe une probabilité ou un niveau de confiance. Le chapitre est organisé de la manière suivante. Après avoir fait un tour d'horizon des différentes techniques utilisées pour la fonction NCTR dans la section 1.1, nous détaillons plus précisement dans la section 1.2 la technique HRD (Haute Résolution Distance) qui est la technique utilisée lors des travaux de thèse. Les différentes méthodes de classification appliquées au contexte NCTR que l'on peut trouver dans la littérature sont détaillées dans la section 1.3. Au final, Nous abordons également le sujet de l'espace de représentation des données dans la section 1.4.

1.1 Fonction NCTR

D'un point de vue technique, la reconnaissance non coopérative peut être abordée par différentes approches : capteurs radar, ESM (Electronic Support Measures), acoustiques, infra-rouges, optroniques, etc... L'approche radar est très attractive parce qu'elle peut être utilisée pour des distances de plusieurs centaines de kilomètres, indifféremment de jour ou de nuit, et elle est relativement insensible aux variations des conditions climatologiques. Le radar est capable de fournir plusieurs paramètres physiques associés à la cible considérée tels que des paramètres cinématiques (altitude, vitesse, accélération, etc...), des paramètres comme la Surface Equivalente Radar (SER) ou la longueur de la cible mais également des paramètres de type spectre Doppler permettant d'analyser les parties mobiles d'une cible.

Pour pouvoir fournir ces informations de natures très différentes, la fonction NCTR repose sur l'exploitation de techniques très différentes. Chacune de ces techniques respecte néanmoins le même processus pour reconnaître une cible à partir de sa signature. En effet, le processus de reconnaissance implique toujours des tâches comme la construction de la base de données, la mesure de la signature de la cible, le conditionnement et l'extraction des attributs et au final l'emploi d'un classifieur sur les attributs et la base de données pour reconnaître la cible.

Les principales techniques NCTR utilisées pour faire de la reconnaissance de cibles résumées dans le tableau 1.1 sont décrites dans cette section.

Techniques NCTR	**Type de signature**	**Attributs extraits**
HRD	Domaine temporel	Profil distance
ISAR 1D/2D	Domaine temporel et fréquence	Image 1D/2D de la cible
JEM	Spectre	Nombre d'aubes moteurs et régimes moteurs

Tab. 1.1 : Différentes techniques NCTR

On peut également citer d'autres techniques NCTR qui ne seront pas détaillées dans la suite du document, comme les techniques basées sur les

paramètres cinématiques d'une cible ou la technique HERM (Helicopter Rotor Modulation) pour les hélicoptères où l'on cherche à récupérer au cours du temps les niveaux d'énergie des pales moteurs et plus précisemment les "flashs" d'énergie.

1.1.1 Inverse Synthetic Aperture Radar (ISAR)

La technique ISAR (Tait 2005, Moruzzis et al. 2004, van Dorp et al. 2012) est équivalente à la technique SAR (Synthetic Aperture Radar) sauf qu'elle utilise le mouvement rotationel de la cible (et les changements d'angle d'aspect) au lieu du mouvement du capteur radar. Cette technique permet d'obtenir une image transverse de la cible. C'est ce qu'on appelle l'ISAR 1D. Les images ISAR 1D sont issues du traitement des spectres Doppler des points brillants (ou diffuseurs) résultant du mouvement de la cible, les points brillants étant définis comme un point ou une surface élémentaire réfléchissant les ondes electromagnétiques. Si la cible tourne en azimut à une vitesse constante sur un petit angle, les points brillants vont se rapprocher ou s'éloigner du radar à une vitesse ne dépendant que de la position transverse de la cible. La rotation entraîne la génération de fréquences Doppler transverses dépendantes qui peuvent être triées par une transformée de Fourier. Cette opération est équivalente à la génération d'une antenne à ouverture synthétique formée par la sommation cohérente des sorties du récepteur.

L'ISAR 1D peut également être associée à la technique HRD pour obtenir une image 2D de la cible. L'axe radial est donnée par la technique HRD et l'axe transverse par la technique ISAR 1D. Cette combinaison des deux techniques NCTR est appelée ISAR 2D.

Il convient de préciser que le temps d'intégration important nécessaire pour obtenir les images ISAR implique l'utilisation de techniques de focalisation sophistiquées afin d'obtenir une image nette.

1.1.2 Jet Engine Modulation (JEM)

La technique JEM (Tong et al. 2005, Tait 2005, Martin and Mulgrew 1990; 1992, Piazza 1999) est une technique de reconnaissance des moteurs à réaction. La technique JEM effectue une analyse Doppler de la signature électromagnétique de la cible, basée sur l'analyse des parties en mouvement à savoir les moteurs et leurs pales. Le spectre JEM est issu de l'interaction du signal radar avec les aubes des différents étages de compression d'un moteur à réaction de l'avion (sous réserve d'une pénétration des ondes dans le conduits d'air, c'est-à-dire d'une longueur d'onde petite). C'est la compréhension de ce phénomène et l'étude des spectres obtenus qui permet de déterminer le nombre de pales et le régime moteur, dans un premier temps, du premier étage de compression et si possible du deuxième étage voire du troisième étage. En utilisant le spectre JEM, plusieurs caractéristiques moteurs peuvent être déterminées : (i) Fréquence de rotation moteur (Composantes Basse fréquence). (ii) "Chopping frequency" (Composantes Haute fréquence) : on analyse dans ce cas chaque étage de compression séparément. Sur le premier étage de compression le

moteur a N pales supposées identiques. Comme le moteur est symétrique par rapport au "spool", à chaque fois qu'il subit une rotation de $1/N$, le même mécanisme se répète. Le signal radar temporel va donc se répéter chaque T/N avec T la période de rotation moteur. La transformée de Fourier de ce signal temporel constitue le spectre JEM pour un étage de compression. Ce spectre est donc caractérisé par des raies séparées par un interval fréquentiel de N/T de la fréquence de transmision du signal radar. Les fréquences de ces raies sont appelées "chopping frequency". Elles permettent de déterminer le nombre de pales multiplié par le nombre de rotations par seconde de chaque étage de compression. (iii) Produits de mélange entre les différents étages : en plus des raies correspondantes aux "chopping frequencies", on retrouve sur le spectre JEM des raies correspondantes aux réflections entre les différents étages de compression.

1.1.3 Définition du taux d'erreur, taux de succès et taux de bonne identification

Les algorithmes détaillés dans la suite du mémoire utilisent une règle de décision binaire (Acceptation/Rejet) ou à trois niveaux de décision (Acceptation/Incertitude/Rejet). Généralement, lorsqu'on utilise une règle de décision binaire, on calcule une mesure de confiance quant à l'appartenance aux différentes classes et on accepte la ou les classes ayant une mesure de confiance élevée (dans le cas contraire, la classe est rejetée). Dans le cas d'une règle de décision à trois niveaux (cf. figure 1.1), pour chaque classe, si la mesure de confiance est très élevée, on accepte la classe, si la mesure de confiance est très faible, on rejette la classe et sinon on déclare la classe incertaine. Les seuils d'acceptation SA et de rejet SR sont ajustés de manière à respecter le taux d'erreur souhaité tout en maximisant le taux de succès.

Il convient ensuite de définir les termes "erreur" et "succès". Il y a **erreur** lorsque la classe i est déclarée **rejetée** pour une signature de la classe i. Il y a **succès** lorsque la classe i est déclarée **acceptée** et les autres classes **incertaines** ou **rejetées** pour une signature de la classe i. On parle de **bonne identification** lorsque la classe i est déclarée **acceptée** et toutes les autres classes **rejetées** pour une signature de la classe i. Le taux de bonne identification sera donc toujours inférieur ou égal au taux de succès. On notera le taux d'erreur TE, le taux de succès TS et le taux de bonne identification TBI dans la suite du mémoire.

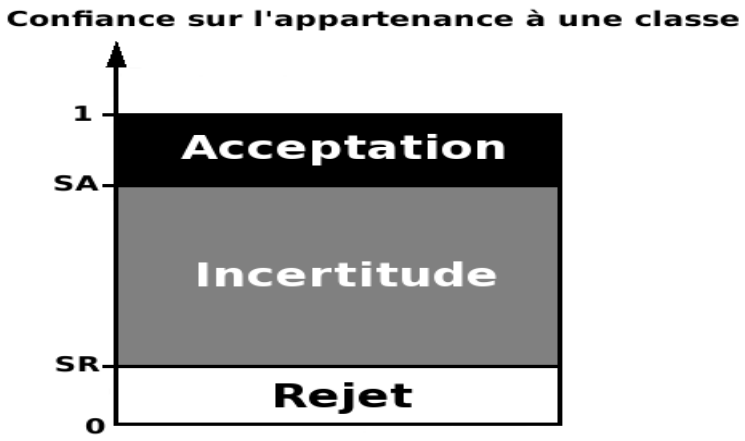


FIG. 1.1 : Règle de décision à trois niveaux

1.2 Haute Résolution Distance (HRD)

La technique Haute Résolution Distance (HRD) offre un moyen simple et rapide de caractériser une cible à travers les profils distance (Wehner and Barnes 1994). Un profil distance est une représentation de la réponse temporelle de la cible à une impulsion radar haute résolution. C'est une image 1D de la cible (cf. figure 1.2). Il a été montré (van der Heiden 1998, Zyweck and Bogner 1996, Hudson and Psaltis 1993, Li and Yang 1993) que les profils distance haute résolution sont des attributs très intéressants et prometteurs pour la classification de cibles, puisqu'ils apportent une information discriminante sur la géométrie de l'avion. C'est la technique retenue pour les travaux de thèse. Dans cette section, nous étudions dans un premier temps les différentes formes d'onde permettant d'obtenir un profil distance HRD (cf. paragraphe 1.2.1) puis nous détaillons dans le paragraphe 1.2.2 la chaîne d'acquisition de ces profils distance. Dans un second temps, nous introduisons la notion de diffuseurs ou de points brillants à l'origine des profils distance (cf. paragraphe 1.2.3) puis nous abordons les sujets de l'influence de l'angle d'aspect (cf. paragraphe 1.2.4) sur les profils distance et de l'alignement de ces derniers (cf. paragraphe 1.2.5).

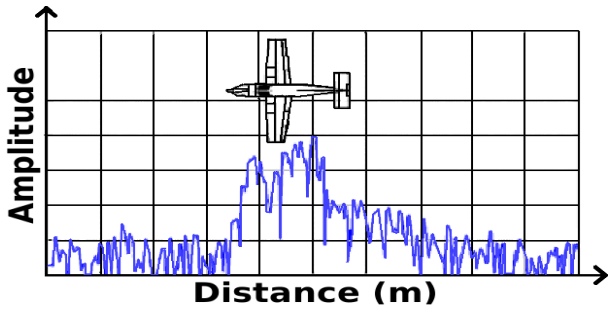


FIG. 1.2 : Un exemple de profil distance

1.2.1 Formes d'onde haute résolution

Il existe plusieurs types de forme d'onde permettant d'obtenir un profil distance. Les paragraphes suivants décrivent ces différentes solutions.

1.2.1.1 Impulsion courte

La solution la plus simple et la plus évidente pour obtenir une forme d'onde haute résolution est d'émettre une impulsion courte. Cependant cette solution a de nombreux inconvénients. Premièrement, l'énergie de l'impulsion est faible car sa durée est très petite. En conséquence, la puissance crête de l'impulsion doit être très importante pour obtenir des performances raisonnables. Un autre inconvénient est le fait que la grande bande passante instantanée associée à une impulsion haute résolution nécessite un convertisseur A/D avec une fréquence d'échantillonnage très importante pour pouvoir enregistrer les échos reçus. Ce type de convertisseur existe mais ils sont souvent chers et consomment beaucoup.

1.2.1.2 Chirp ou forme d'onde modulée linéairement en fréquence

Une solution pour augmenter l'énergie de l'impulsion tout en maintenant une haute résolution est d'appliquer une forme d'onde dite "chirp" qui est une onde sinusoïdale modulée linéairement en fréquence. La durée de l'impulsion dépend du rapport signal sur bruit (RSB) souhaité. Le "chirp" utilise le principe de compression d'impulsion qui permet d'augmenter la résolution en distance et le RSB, par modulation du signal émis.

La fréquence $f(t)$ d'un chirp peut donc s'écrire

$$f(t) = f_0 + \alpha t, \tag{1.1}$$

où f_0 est la fréquence de départ et α le taux de chirp [Hz/s] ou plus couramment "chirp rate". Puisque la fréquence est réliée à la phase par la relation

$$f(t) = \frac{1}{2\pi}\frac{\mathrm{d}\phi}{\mathrm{d}t}, \tag{1.2}$$

la phase de la forme d'onde *chirp* peut s'écrire de la manière suivante :

$$\phi(t) = 2\pi \int_0^t f(\tau)\mathrm{d}\tau = 2\pi(f_0 t + \frac{\alpha}{2}t^2). \tag{1.3}$$

En utilisant cette expression, un *chirp* de durée T_c peut s'écrire sous la forme exponentielle suivante :

$$p(t) = e^{j(\omega_0 t + \pi\alpha t^2)},\ 0 < t < T_c, \tag{1.4}$$

La figure 1.3 illustre le comportement temps-fréquence d'un *chirp*.

Comme pour l'impulsion courte, l'inconvénient du chirp est qu'il nécessite un convertisseur A/D avec une fréquence d'échantillonnage très importante pour pouvoir enregistrer les échos reçus.

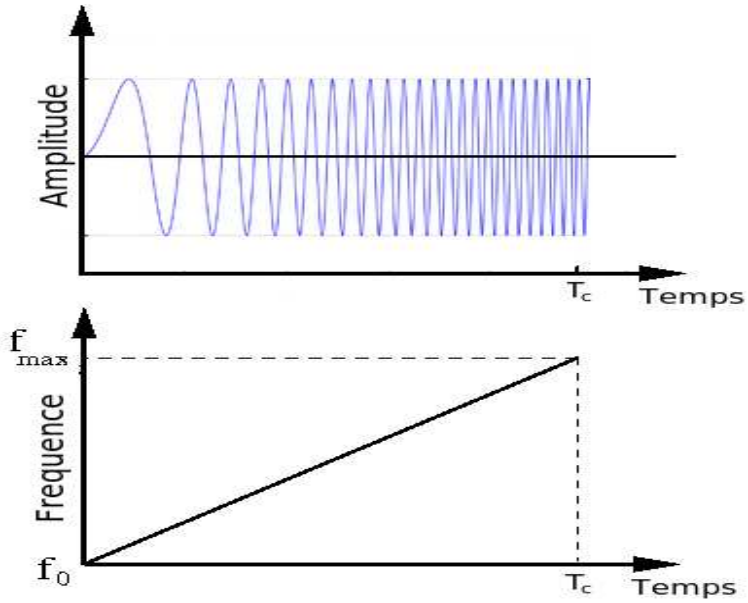


FIG. 1.3 : Forme d'onde "Chirp"

1.2.1.3 Stepped Frequency ou bande synthétique

Une autre solution équivalente au "chirp" est la technique dite "stepped frequency". Cette technique est implémentée en augmentant la fréquence d'émission radar d'un pas Δf de f_1 jusqu'à f_N. Δf est égale à la bande passante totale B divisée par le nombre de pas en fréquence. On peut l'écrire de la manière suivante :

$$\Delta f = \frac{B}{N_f - 1} \tag{1.5}$$

Si N_f fréquences sont utilisées, N_f mesures d'amplitude et de phase sont obtenues ce qui correspond à N_f nombres complexes. On obtient alors le spectre fréquentiel de la cible.

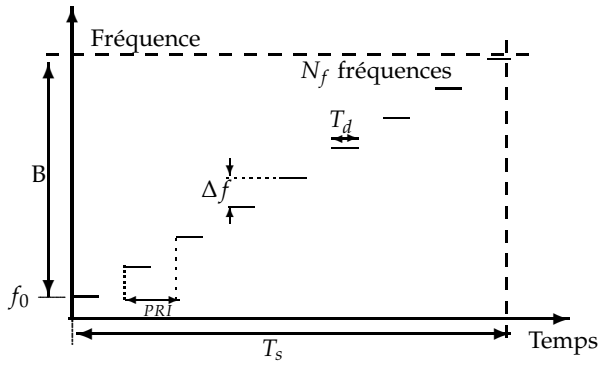


FIG. 1.4 : Forme d'onde "Stepped-frequency"

où le temps d'illumination T_d représente le temps passé sur chaque fréquence et PRI le temps entre deux impulsions. La durée totale de la forme d'onde est :

$$T_s = N_f PRI \tag{1.6}$$

Par rapport aux chirps ou aux impulsions courtes, l'avantage avec cette forme d'onde est qu'il n'y a pas besoin d'un convertisseur rapide car la numérisation est faite impulsion par impulsion, la largeur de bande des impulsions étant faible. Par contre la durée de la forme d'onde est plus importante.

1.2.1.4 Stretch

La méthode "Stretch" consiste à envoyer une forme d'onde modulée linéairement en fréquence et à démoduler les échos de retour en les mélangeant à un signal modulé en fréquence ayant une pente identique de la forme d'onde transmise. La figure 1.5 illustre le principe de cette méthode sur un exemple d'une cible avec trois points brillants.

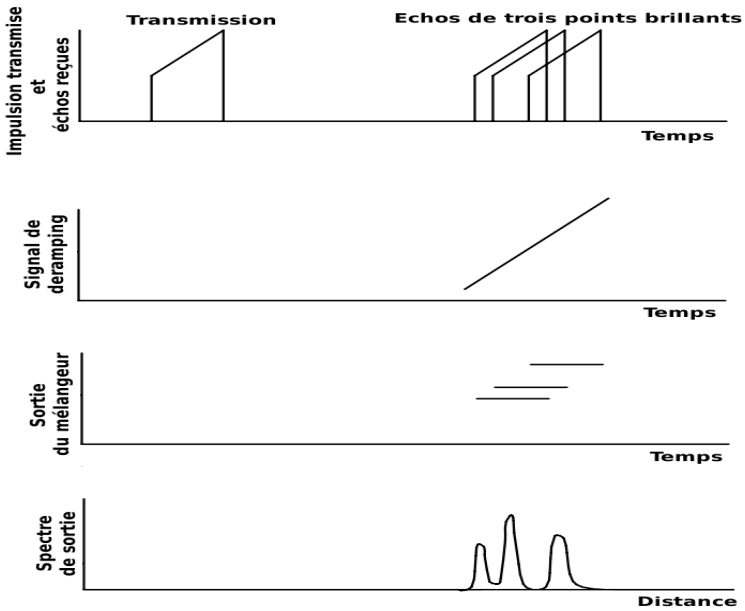


FIG. 1.5 : Forme d'onde stretch

L'avantage de cette technique est qu'il n'y a pas besoin d'un convertisseur A/D complexe car la largeur de bande de numérisation a été réduite grâce au deramping analogique. L'inconvénient est que la fenêtre d'acquisition est réduite à quelques kilomètres car le processus ne peut être effectué qu'une seule fois.

1.2.2 Obtention d'un profil distance HRD

D'une manière générale et cela quelle que soit la forme d'onde transmise par le module de transmission (TX) du radar, la chaîne d'acquisition des profils distance se compose d'un récepteur IQ (In-phase and Quadrature) pour transformer l'onde radar en un signal fréquentiel complexe et d'un bloc permettant de calculer la transformée de Fourier de ce signal (cf.

figure 1.6). En effet, le signal acquis par le récepteur (RX) est un signal fréquentiel représentant la valeur de la surface équivalente radar (SER) pour chaque fréquence émise. Avant toute chose, le signal est pondéré par une fenêtre (le plus souvent de Hamming), de manière à limiter l'influence des lobes secondaires. Ensuite pour obtenir un profil distance, la transformée de Fourier de ce signal fréquentiel pondéré est calculée de manière à obtenir une représentation de l'amplitude du signal radar en fonction de la distance. Finalement, le profil distance est obtenu en prenant le module de ce signal [1]. Le profil distance est le plus souvent exprimé à l'échelle logarithmique de manière à faire ressortir les points brillants de faible amplitude.

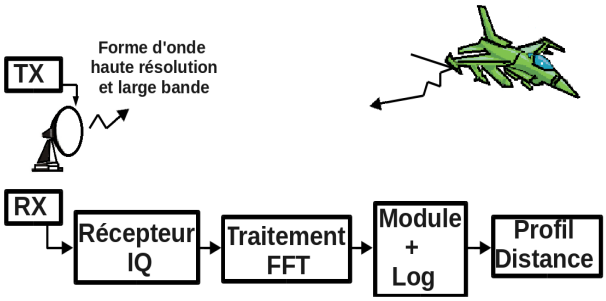


Fig. 1.6 : Formation du profil distance

1.2.3 Diffuseurs ou points brillants

Par approximation, on peut considérer que les profils distance sont le résultat d'un certain nombre de diffusions indépendantes. On considère qu'une cible est constituée de plusieurs diffuseurs (cf. figure 1.7). On parle également de points brillants que l'on peut définir comme un point ou une surface élémentaire réfléchissant les ondes électromagnétiques.

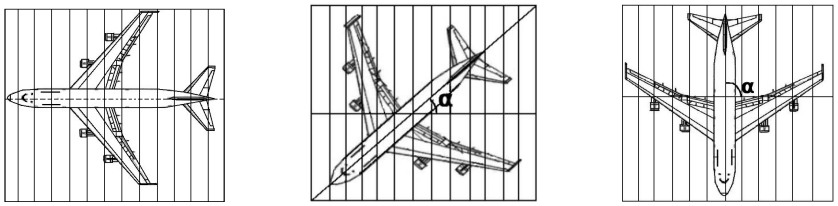
[1] On est conscient qu'en prenant le module, on perd l'information de phase contenue dans le signal complexe. Pour pouvoir travailler sur les signaux complexes, il auraient fallu en amont réaliser une étude mettant en évidence le pouvoir discriminant de l'information contenue dans la phase puis une autre étude explicitant précisément la nature de l'information contenue dans la phase. Ces deux axes de recherche peuvent potentiellement définir un sujet de thèse à part entière et l'objectif de la thèse n'était pas celui-là.

FIG. 1.7 : Un exemple de diffuseurs sur une cible

Les réflexions sur ces diffuseurs proviennent des différentes structures physiques de la cible. Par conséquent, il existe des phénomènes d'interférences constructives et destructives entre les diffuseurs. La combinaison des diffuseurs et les interférences entre ces différents diffuseurs rendent la modélisation de la diffusion très complexe et il est souvent difficile de prédire efficacement un profil distance à partir d'un modèle de diffusion simple (Rihaczek and Herschowitz 2000).

1.2.4 Influence de l'angle d'aspect

L'effet principal de cette combinaison est que les profils distance peuvent varier très rapidement suivant l'angle d'aspect. L'angle d'aspect d'un avion peut être exprimé comme une paire de coordonnées (α, θ) où α représente l'azimut et θ l'angle de site. L'azimut est l'angle entre la direction du nez de l'avion et la direction de la ligne de vue du radar projetée sur le plan formé par le nez et les ailes de l'avion. L'angle de site est l'angle entre la ligne de vue du radar et le plan formé par le nez et les ailes de l'avion. Suivant les valeurs de l'azimut α et de l'angle de site θ, les parties de l'avion contribuant au profil distance varient. Cela est illustré sur la figure 1.8. Sur la figure 1.8a, les cases distance sont distribuées suivant l'axe du fuselage de l'appareil. Dans cette configuration, on peut clairement identifier quelles parties de l'avion contribuent au profil distance. Sur la figure 1.8b, on se rend compte que les ailes de l'avion contribueront à la valeur du signal de retour en plus du fuselage pour la plupart des cases distance. Pour la figure 1.8c, le fuselage est restreint à quelques cases distance et la reconnaissance s'effectue sur la largeur de l'appareil



(a) (b) (c)

FIG. 1.8 : Influence de l'angle d'aspect

De nombreux articles reviennent sur les propriétés des profils distance et également sur leur sensibilité à l'angle d'aspect. L'article de Mengdao (Mengdao and Zheng 2001) revient sur les principales caractéristiques des profils distance d'avions. Les propriétés des échos de la cible peuvent être calculées à partir des propriétés de diffusion de la cible mais cela est très compliqué. On adopte donc dans la plupart des cas des modèles où les diffuseurs sont des points et où la cible est vue comme un ensemble de diffuseurs discrets se trouvant sur la surface éclairée par le radar. L'article traite ensuite plus précisemment de la sensibilité des profils distance notamment à l'angle d'aspect et propose une méthode pour traiter ce problème. Selon les auteurs de l'article, le modèle de distribution d'un diffuseur est lié à l'angle de vue mais il change très lentement. Cependant le profil distance est le résultat de la projection de nombreux diffuseurs sur la ligne de vue du radar. Cette projection est obtenue en faisant la somme vectorielle des échos des différents diffuseurs, c'est-à-dire la somme des amplitudes complexes. La phase des différents sous-échos de la cible est sensible au déplacement radial, ce qui rend très sensible les profils distance aux changements d'angle d'aspect. Un changement d'angle d'aspect cause deux effets : le premier est la variation de la différence de phase et l'autre est le déplacement de l'enveloppe (décalage en distance) de la cible. Si le déplacement de l'enveloppe est plus grand que la résolution distance, la distribution des diffuseurs dans chaque case distance est modifiée. Ce phénomène est appellé Migration Through Resolution Cells (MTRC) par les auteurs. Pour éviter ce phénomène, l'angle de rotation doit être restreint, c'est-à-dire si la résolution distance est notée ΔR et la cible est de taille L, l'angle de rotation doit être restreint à $\delta\phi \leq \Delta R/L$. Si on prend une bande passante de $400MHz$ et une cible de 15 mètres, $\delta\phi$ doit être plus petit que 1.5°. Les auteurs de l'article étudient ensuite les propriétés d'un profil distance dans le cas où le $\delta\phi$ respecte la condition décrite juste avant. Les auteurs s'intéressent notamment aux mouvements de l'avion. Les auteurs décomposent ce mouvement en deux parties différentes : le mouvement de translation et le mouvement de rotation. Lors d'une translation, la posture de la cible par rapport au radar n'est pas changée, la forme du profil distance est donc inchangée. Seule l'enveloppe de la cible est décalée. Quand la cible subit une rotation, lorsque qu'on se place dans les conditions permettant de ne pas avoir de MTRC, la distribution des diffuseurs dans chaque case distance est inchangée. En supposant qu'il y a L_n diffuseurs dans la n-ème case distance, à cause de la rotation, un déplacement radial est observé. On note $\Delta r_i(m)$ le déplacement radial du i-ème diffuseur dans le m-ème écho. On peut donc définir le m-ème écho de la n-ème case distance de la manière suivante :

$$x_n(m) = \sum_i^{L_n} \sigma_i e^{-j(\frac{4\pi}{\lambda}\Delta r_i(m) - \psi_{i0})} = \sum_i^{L_n} \sigma_i e^{j\phi_{ni}m} \tag{1.7}$$

L'expression de la phase est donc :

$$\phi_{ni}m = -\frac{4\pi}{\lambda}\Delta r_i(m) + \psi_{i0} \tag{1.8}$$

La puissance de $x_n(m)$ est :

$$|x_n(m)|^2 = x_n(m)x_n^\star(m) = \sum_{i=1}^{L_n} \sigma_i^2 + 2\sum_{i=2}^{L_n}\sum_{k=1}^{i} \sigma_i\sigma_k\xi_{nik}(m) \tag{1.9}$$

où :

$$\xi_{nik}(m) = cos(\theta_{nik}(m)) \tag{1.10}$$

et

$$\begin{aligned} \theta_{nik}(m) &= \phi_{ni}(m) - \phi_{nk}(m) \\ &= (\psi_{i0} - \psi_{k0}) - \frac{4\pi}{\lambda}(\Delta r_i(m) - \Delta r_k(m)) \\ &= \theta_{nik}(0) + \delta\theta_{nik}(m) \end{aligned} \tag{1.11}$$

$\theta_{nik}(m)$ est la différence de phase entre le i-ème et le k-ème diffuseur dans la n-ème case distance pour le m-ème écho.

D'après ces équations, on remarque que la puissance de l'écho de la cible est composée de deux termes, le premier étant la somme de l'intensité des différents diffuseurs (qui ne dépend pas de la rotation dans le cas où on respecte les conditions pour ne pas avoir de MTRC) et le second étant le terme croisé entre les différents diffuseurs. L'article propose d'étudier les propriétés statistiques de ce terme croisé $\xi_{nik}(m)$.

D'après les auteurs, si la valeur maximale de $\delta\theta_{nik}(m)$ est plus petite que $\pi/2$ alors la variation du terme croisé $\xi_{nik}(m)$ sera petite. Les auteurs voient ce terme comme un processus aléatoire dont la moyenne est nulle et dont l'angle de corrélation est de l'ordre d'une centaine de degrés.

Une des méthodes proposées pour traiter ce problème de variabilité du profil distance en fonction de l'angle de rotation (c'est-à-dire rendre le terme croisé négligeable) consiste à moyenner une dizaine de profils distance ayant un angle de vue compris dans un certain intervalle angulaire. Une alternative à ce moyennage est de réaliser une analyse en composantes principales sur les profils distance pour lesquels l'angle d'aspect a varié.

Au final, les auteurs proposent de classer la distribution des diffuseurs en trois classes :

- La première classe est celle où il y a un seul diffuseur principal pour une case distance, les autres diffuseurs faibles pouvant être vus comme du bruit. L'amplitude de l'écho dans cette case distance est déterminée par celle du diffuseur principal.

- La deuxième classe est celle où il n'y a pas de diffuseurs prédominants mais un nombre important de petits diffuseurs. L'amplitude dans cette case distance ondule et peut être vue comme une distribution de Rayleigh.
- La troisème classe est celle où seulement quelques diffuseurs prédominants avec plusieurs diffuseurs faibles sont présents. S'il y a deux diffuseurs prédominants, quand la cible tourne, l'amplitude ondule sérieusement à cause de la différence de phase entre les échos issus des deux diffuseurs.

On retrouve cette même découpe dans l'article de Du (Du and al. 2006), mais avec un peu plus de précision sur les distributions des trois différentes classes. Dans l'article de Du, la première classe de distribution d'amplitude de l'écho reçu dans une case distance peut être vue comme une distribution de Rice. La deuxième est également vue comme une distribution de Rayleigh. Pour la troisième classe, elle peut être vue comme une distribution multimodale (principalement bimodale) qui dépend de la distribution des diffuseurs prédominants.

1.2.5 Alignement des profils distance HRD

L'article de Zwart et al. (Zwart et al. 2003) revient sur cette sensibilité des profils distance à l'angle d'aspect, phénomène que les auteurs ont appelé *speckle*. Cependant l'article s'intéresse plus précisement au phénomène appelé par les auteurs *Translational Range Migration*. Ce phénomène classique lorsqu'on travaille avec les profils distance haute résolution nous oblige à réaligner les profils distance d'une base de données avant de pouvoir les comparer. En effet, la localisation des diffuseurs dans un profil distance dépend de la distance entre le radar et l'avion. Or cette distance n'est pas connue de manière suffisamment précise pour que les profils distance soient correctement alignés. On entend par alignement (ou recalage), la procédure qui cherche à placer le nez de l'appareil à la même position sur tous les profils distance. L'article revient donc sur les méthodes classiques d'alignement des profils distance. Les auteurs distinguent deux types de procédure d'alignement : l'alignement relatif et l'alignement absolu.

L'alignement relatif (Kosir et al. 1995) correspond à la procédure qui permet d'aligner un profil distance avec un ou plusieurs autres profils distance d'une base de données. L'alignement relatif est communément réalisé en alignant une paire de profil distance de manière à maximiser leur corrélation.

L'alignement absolu (Liao and Bao 1998) signifie quant à lui que les profils distance sont récalés pour optimiser un certain critère d'optimisation extérieur (c'est-à-dire ne dépendant pas des autres profils distance).

Une autre approche citée dans l'article pour traiter ce problème d'alignement des profils distance est de travailler avec le module de la transformée de Fourier des profils distance. Cette approche est également abordée dans le livre de François Le Chevalier (Chevalier 2002) ou encore dans les articles de Zyweck et Bogner (Zyweck and Bogner 1996) ou de Wong (Wong 2004). Le principal désavantage de cette méthode est que l'on perd l'information de phase, information potentiellement discrimi-

nante, ce qui peut donc faire baisser les performances de classification. L'information de distance étant contenue dans la phase, il n'est au final plus nécessaire de recaler les profils distance lorsqu'on travaille avec le module de la transformée de Fourier. Nous reviendrons un peu plus tard sur l'étude des techniques NCTR dans le domaine fréquentiel (cf. paragraphe 1.4.1).

Après être revenu sur différentes méthodes d'alignement des profils distance, l'article (Zwart et al. 2003) propose une nouvelle méthode pour résoudre le problème de l'alignement en distance des profils distance. Les auteurs l'ont appellé *Time-Smoothed Zero Phase Representation*. Les auteurs l'ont concu pour combiner les avantages de l'alignement relatif avec les avantages de l'alignement absolu.

Pour n'importe quelle fonction $f(x)$ dont la transformée de Fourier est $F(f)$, la transformée de Fourier de $f(x-s)$ est donnée par :

$$F(f(x-s)) = F(f(x))e^{i\omega s} \tag{1.12}$$

c'est-à-dire par un décalage de phase de $F(f(x))$. Dans le cas discret, on travaille avec la fonction échantillonnée f_n de f avec $n = 1 \dots d$, on peut écrire le même type de relation :

$$F(f_{(n+k) \bmod \mathrm{d}}) = F(f_n)e^{2\pi ink/d} \tag{1.13}$$

Pour un décalage discret de k, la phase ϕ de la première composante continue de la transformée de Fourier de f_n sera décalée de $2\pi k/d$ et les phases des composantes d'ordre supérieur ajustées en conséquence.

Si on suppose que l'on a une série de N profils distance $\boldsymbol{x}_i$ avec $i = 0, \dots, N-1$ correspondant à un avion en vol à différents moments t_i. Si on calcule pour chacun de ces profils distance, la phase de la première composante continue de la transformée de Fourier, on obtient une séquence $\phi(t_i) = \phi_i$.

Les variations de ϕ_i sont causées par des rotations ou des translations de la cible par rapport au radar. Supposons que la cible reste à une distance R du radar durant la rotation. ϕ_i est donc une fonction uniquement dépendante de l'angle d'aspect (α, θ). Dans ce cas, on peut donc définir ϕ_i de la manière suivante :

$$\phi_i = \phi(\alpha(t_i), \theta(t_i)) = \phi_i^a \tag{1.14}$$

Cependant dans le cas réel, la distance radar-cible r_i varie également dans le temps. On peut donc réécrire ϕ_i de la manière suivante :

$$\phi_i = \phi_i^a + 2\pi/L(r_i - R) = \phi_i^a + \gamma(r_i - R) \tag{1.15}$$

avec L la taille totale de la fenêtre distance (en mètre).

Si l'on connaissait parfaitement (c'est-à-dire sans erreur) la distance r_i entre la cible et le radar une manière de compenser les effets translationnels serait tout simplement de soustraire la quantité $\gamma(r_i - R)$ à la phase ϕ_i. Dans le cas des données utilisées pour la thèse les distances sont généralement estimées. Les r_i sont donc connus mais avec des erreurs. Il

est donc utile de s'intéresser au cas où les r_i sont inconnus.

Dans le cas où les r_i sont inconnus, une première solution, appelée représentation *Pure Zero Phase* par les auteurs, consiste tout simplement à mettre chaque ϕ_i à 0 (et d'ajuster la phase des autres composantes d'ordre supérieur en conséquence). Cela a pour effet d'éliminer les effets translationnels introduits par le terme $\gamma(r_i - R)$ mais cela introduit en même temps une erreur d'alignement due à la non prise en compte du terme ϕ_i^a.

Les auteurs proposent donc une méthode plus sophistiquée pour traiter le terme de distance $\gamma(r_i - R)$. Il s'agit de la méthode *Time-Smoothed Zero Phase Representation*. Cette méthode est basée sur l'hypothèse que la position relative de la majorité des diffuseurs principaux dans le profil distance reste stable lorsque l'angle d'aspect varie très faiblement. Deux profils distance successifs dans une base de données peuvent donc être recalés entre eux en utilisant une méthode de recalage relatif telle que celle du maximum de corrélation. Le but de cette nouvelle méthode est de combiner les avantages du recalage relatif à travers le calcul de la corrélation maximale et les avantages du recalage absolu en utilisant la *Pure Zero Phase Representation*.

A partir de l'équation 1.15, on peut écrire la relation entre les phases ϕ_i et ϕ_{i-1} suivante :

$$\phi_i - \phi_{i-1} = \phi_i^a - \phi_{i-1}^a + \gamma(r_i - r_{i-1}) = \phi_i^a - \phi_{i-1}^a + \gamma\Delta(r_i) \tag{1.16}$$

Pour chaque profil distance $\boldsymbol{x}_i$, on peut donc déterminer le décalage optimal par rapport au profil distance précédent $\boldsymbol{x}_{i-1}$. Ce décalage correspond à un décalage en phase $\delta\phi_i^c$. Dire que corriger ce décalage de phase permet d'aligner parfaitement les profils $\boldsymbol{x}_i$ et $\boldsymbol{x}_{i-1}$ est équivalent à dire que $\delta\phi_i^c = \gamma\Delta r_i$. Comme γ est connu, pour chaque profil distance à part le premier, on a une estimation de Δr_i.

On peut donc corriger la phase d'origine ϕ_i par la phase ϕ_i^c :

$$\phi_i^c = \phi_i^a + \gamma(r_i - R - \sum_{j=1}^{i} \Delta r_j) = \phi_i^a + \gamma(r_0 - R) \tag{1.17}$$

En d'autres termes, avec cette phase corrigée, tous les profils distance sont maintenant recalés à la distance $r = r_0$ du radar.

Pour obtenir, un recalage absolu, tous les profils distance doivent être recalés à la distance de référence R. Cela peut être fait si r_0 est connu, or r_0 n'est pas connu, les auteurs proposent une méthode heuristique pour l'estimer. A chaque t_i, $\gamma(r_0 - R)$ est estimé localement par une moyenne mobile sur ϕ_i^c. Cette estimation est ensuite soustraite de ϕ_i^c pour obtenir une phase finale estimée ϕ_i^f.

$$\phi_i^f = \phi_i^c - \frac{1}{w+1} \sum_{k=-w}^{w} \phi_{i+k}^c \tag{1.18}$$

Cette phase corrigée nous permet d'interpoler notre procédure d'alignement entre l'alignement absolu ($w = 1$) et l'alignement relatif ($w = N$). En théorie, la taille de la fenêtre optimale dépend de l'angle d'aspect des profils distance successifs. Pour des petites différences d'angle d'aspect, l'alignement par maximum de corrélation marche bien et on peut donc utiliser de grandes valeurs pour w. Quand, dans la base de données, il y a de grandes variations d'angle d'aspect entre profils distance successifs, on doit plutôt utiliser de petites valeurs pour w. Les auteurs précisent néanmoins que la valeur de w a été fixée de manière empirique pour leur expérience.

La méthode SZPR est plutôt bien adaptée au cas où l'on doit aligner l'ensemble des profils distance dans une base de données. Avec nos données, cela n'est pas le cas. En effet, les profils distance de la base d'apprentissage sont déjà alignés entre eux (cf. chapitre 2, section 2.1.1). Ils ont été alignés par une méthode de type maximum de corrélation. Cela se justifie par le fait que la plage de variation des angles d'aspect des profils distance de la base d'apprentissage est restreinte à quelques degrés et la variation d'angle d'aspect entre deux profils distance successifs n'est que de $0.25°$. Sous ces conditions, l'alignement relatif est la meilleure solution pour aligner les profils distance de la base d'apprentissage.

La base de test, elle, n'est pas recalée. Cependant, cela ne nous intéresse pas de recaler les profils distance de cette base de test entre eux. En effet, dans le cas opérationnel, dès qu'un profil distance est mesuré par le radar, on exécute l'algorithme de reconnaissance sur cet unique profil distance sans attendre de savoir si d'autres mesures de la cible seront disponibles par la suite. Pour schématiser, cela veut dire qu'en situation opérationnelle, la base de test ne sera constituée que d'un seul profil distance. La problématique est plutôt de recaler le profil distance de test avec les profils distance de la base d'apprentissage. Théoriquement, étant donnée que la base d'apprentissage est déjà recalée, il suffirait d'aligner le profil distance de test par rapport à un seul des profils distance d'apprentissage pour qu'il soit aligné avec la base de test. Dans la pratique, et cela sera le cas pour les algorithmes présentés dans les chapitres 3, 4 et 5, on préfèrera aligner le profil distance de test à chaque profil d'apprentissage. La première raison est que si un profil distance d'apprentissage n'est pas bien aligné avec les autres, on peut se retrouver à avoir le profil distance de test mal aligné par rapport à tous les profils d'apprentissage. De plus, si un profil distance est mal aligné dans la base d'apprentissage par rapport au profil distance de référence, il se peut qu'il soit bien aligné lorsqu'on calcule le maximum de corrélation avec le profil distance de test. Il vaut donc mieux aligner le profil distance de test avec chaque profil distance d'apprentissage. Cela correspond donc à un alignement de type relatif. Avec les algorithmes présentés dans les chapitres 3, 4 et 5, le fait d'avoir la base d'apprentissage déjà recalée peut se voir comme un moyen de restreindre la fenêtre de décalage du profil distance de test à quelques échantillons.

Dans le cas des approches probabilistes, la problématique n'est pas tout à fait la même (cf. chapitre 4). En effet, on va chercher à modéliser les classes par un modèle (ce qui n'est pas le cas avec les KPPV et les algorithmes basés sur la logique floue). Les paramètres de ce modèle étant

estimés à partir de la base d'apprentissage, pour avoir un modèle le plus précis possible, il faut que les profils distance d'apprentissage d'une même classe soient recalés entre eux. Mais il faut aussi que ces profils distance d'apprentissage soient recalés avec le profil distance de test qui vient d'être mesuré par le radar. Encore une fois, nous utiliserons l'alignement relatif pour recaler tous les profils distance d'apprentissage au profil distance de test. On considère que lorsque les profils distance d'apprentissage sont recalés avec le profil distance de test, ils sont recalés entre eux et on peut donc estimer les paramètres des différentes classes à partir de ces profils distance d'apprentissage recalés.

1.3 Méthodes de classification

Quand on parle de méthodes de classification, on peut distinguer deux grands types d'approches :
- l'approche non paramétrique
- l'approche paramétrique.

L'*approche non paramétrique* considère une seule hypothèse, celle de dire que plus deux individus sont proches et plus ils ont de chances d'appartenir à la même classe.

L'*approche paramétrique* utilise la distribution statistique des attributs pour faire la classification. La forme de ces distributions dépend d'un certain nombre de paramètres, d'où le terme d'*approche paramétrique*. Les *approches probabilistes*, par exemple, sont des *approches paramétriques*. Avec ce type d'approches, on peut par exemple considérer que les individus de chacune des classes suivent une loi normale. Le problème qui se pose avec ce type d'approches consiste à déterminer quels sont les paramètres des lois et à quelles classes chaque individu a le plus de chance d'appartenir.

On peut également faire une distinction plus générale parmi les méthodes de classification en regardant le mode d'apprentissage utilisé. Il existe deux grandes méthodes d'apprentissage :
- l'apprentissage supervisé
- l'apprentissage non supervisé.

On parle de méthodes de classification *supervisées* lorsque les classes des échantillons de la base d'apprentissage sont connues. On trouve également dans la littérature le terme d'*analyse discriminante*. Les méthodes que nous utiliserons dans le cadre de la thèse seront des méthodes *supervisées*. Concrètement, cela signifie que notre base d'apprentissage contient des profils distance dont la classe est connue et une base de test qui nous permettra de tester les performances de nos méthodes de classification.

On parle de méthode de classification *non supervisée* lorsque la base d'apprentissage est composée d'échantillons dont la classe est inconnue et lorsque le nombre de classes est également inconnu.

Outre ces deux principaux types d'apprentissage, on peut définir deux autres types d'apprentissage :

– l'apprentissage semi-supervisé
– l'apprentissage partiellement-supervisé.

Les méthodes d'apprentissage *semi-supervisées* sont des méthodes utilisées lorsque certaines données ou étiquettes sont manquantes. On parle de méthodes de classification *partiellement-supervisées* lorsque pour certains échantillons de la base d'apprentissage, on ne renseigne pas une classe d'appartenance précise mais plutôt une appartenance à plusieurs classes possibles.

Il existe de nombreux livres sur ce sujet. Le livre de Duda, Hart et Stork (Duda et al. 2001) souvent cité comme référence dans de nombreux articles permet de retrouver en détail une grande partie des méthodes de classification existantes. Le livre de Webb (Webb 2002), plus axé sur les méthodes statistiques, est également un très bon livre qui passe en revue et détaille un grand nombre de méthodes de classification.

Durant cette thèse, nous nous sommes intéressés tout particulièrement à trois types de méthodes :
– les K plus proches voisins (cf. chapitre 3)
– les méthodes probabilistes (cf. chapitre 4)
– les méthodes basées sur la logique floue (cf. chapitre 5).

Dans la suite de cette section, en nous appuyant sur les articles existant dans la littérature, nous nous focalisons sur les méthodes de classification appliquées aux profils distance haute résolution. Dans le paragraphe 1.3.1, nous présentons les méthodes de classification non paramétrique puis nous abordons dans le paragraphe 1.3.2 les méthodes de classification paramétrique parmi lesquelles les méthodes bayésiennes, les machines à vecteur de support ou encore les méthodes basées sur la logique floue.

1.3.1 Classification non paramétrique

Les méthodes non paramétriques consistent à estimer la densité de probabilités directement à partir des données. Aucune hypothèse *a priori* sur la distribution des données n'est posée. Les trois approches principales sont l'estimation par histogramme, la méthode de la fenêtre de Parzen (Parzen 1962) et la méthode des K Plus Proches Voisins (KPPV). On peut retrouver plus en détail ces différentes approches dans le livre de Bishop (Bishop 2006).

1.3.1.1 Estimation par histogramme

L'estimation par histogramme consiste à diviser l'espace d'entrée en compartiment de taille égale h. On construit l'histogramme en comptant le nombre d'échantillons dans chacun des compartiments. Plus la taille h des compartiments est petite et plus l'estimation de la densité sera fine. Lorsque la densité à estimer est de dimension multiple, les compartiments correspondent à des hypercubes d'hypervolumes égaux. Cette méthode est applicable sur des jeux de données de dimension faible. Pour l'estimation de densité de probabilité de grande dimension, la méthode souffre

gravement de la "malédiction de la dimension" ou "curse of dimensionality" en anglais. De plus, pour que l'estimation converge vers les véritables densités de probabilité trois conditions doivent être respectées :

- Le volume de chaque compartiment doit être réduit ;
- Le nombre d'observations par compartiment doit être très grand ;
- Le ratio du nombre d'observations par compartiment avec le nombre total d'observations doit être élevé.

Lorsqu'un nouvel échantillon est disponible pour l'apprentissage, il est affecté à un des compartiments de l'histogramme, la densité estimée est donc modifiée. Une fois que les densités de probabilités affectées à chaque classe sont estimées, la classification suit le schéma classique qui consiste à calculer la probabilité qu'a un nouvel échantillon d'appartenir à l'une des classes (modélisées par sa densité de probabilité). L'échantillon sera affecté à la classe ayant la plus grande probabilité.

Cette méthode est très difficilement applicable à des problèmes où la dimension des données est grande car pour construire l'histogramme, le nombre d'observations doit être très important (ce qui n'est pas toujours possible) et dans ce cas la charge de calcul devient vite importante.

1.3.1.2 Fenêtre de Parzen

Il s'agit d'une estimation plus douce que l'estimateur par histogramme. Avec l'estimateur par histogramme, la densité en un point x est estimée par la proportion d'observations $x_1, \ldots, x_N$ qui se trouvent à proximité de x. Pour cela, on trace une boîte en x dont la largeur est gouvernée par un paramètre de lissage h. On compte ensuite le nombre d'observations qui appartiennent à cette boîte. Cette estimation, qui dépend du paramètre de lissage h, présente de bonnes propriétés statistiques mais est par construction non-continue.

La méthode du noyau consiste à récupérer la continuité : pour cela, on remplace la boîte centrée en x et de largeur h par une courbe en cloche centrée en x. Plus une observation est proche du point de support x plus la courbe en cloche lui donnera une valeur numérique importante. A l'inverse, les observations trop éloignées de x se voient affecter une valeur numérique négligeable. L'estimateur est formé par la somme (ou plutôt la moyenne) des courbes en cloche. Cet estimateur est donc clairement continu. Dans la pratique cela se résume à convoluer les données par un noyau gaussien. Dans le cas multidimensionnel, le noyau gaussien est un noyau gaussien multivarié. Autant dire encore une fois, que cette méthode est sensible à la dimensionnalité du problème, ce qui la rend très difficile à utiliser dans de nombreux cas.

1.3.1.3 K plus proches voisins (KPPV)

Le principe de l'algorithme des KPPV est repris pour un des algorithmes présentés dans ce mémoire (cf. chapitre 3). Le principe des KPPV est le suivant : on dispose de N échantillons dont on connaît la classe (base d'apprentissage). Pour chaque nouvel échantillon, on détermine

la classe à laquelle il appartient. La méthode des KPPV va prendre en compte de manière identique les K échantillons de la base d'apprentissage les plus proches de l'échantillon à classer. On affecte à l'échantillon à classer la classe revenant majoritairement parmi les K plus proches échantillons d'apprentissage. On peut voir cette méthode comme une méthode d'estimation de la densité adaptant la largeur de la fenêtre selon la densité locale des données (K données les plus proches). Il s'agit en fait de l'estimateur par histogramme où la taille h des compartiments est adaptative.

On retrouve peu d'algorithmes de reconnaissance de cibles dans la littérature basés sur l'algorithme des KPPV, du fait de la charge de calcul bien plus importante qu'avec d'autres méthodes, bayésiennes par exemple. Cependant, en 2009, Mingjing et al. (Mingjing et al. 2009), propose d'évaluer les performances d'un classifieur KPPV sur de nouveaux attributs extraits des profils distance haute résolution. La nouveauté de l'article réside plutôt dans le type des attributs utilisés que dans le choix du classifieur. Il s'agit en fait d'une combinaison de deux types d'attributs invariants aux translations des profils distance. Les auteurs proposent, en effet, de combiner les moments centraux et la distribution d'entropie extraits des profils distance pour former un seul vecteur d'attributs. Les profils distance subissent dans un premier temps une série de prétraitements. Le premier traitement est d'aligner les profils distance entre eux en utilisant la corrélation croisée. Les auteurs préconisent également de moyenner plusieurs profils entre eux de manière à augmenter le RSB. En effet, d'après les auteurs si M profils distance sont moyennés alors l'augmentation de RSB sera de $\sqrt(M)$. Dans notre cas, où on traitera un seul profil distance à la fois, cela n'est pas applicable. Le dernier traitement consiste à normaliser les profils distance. Si on note $\boldsymbol{x}$ un profil distance avec N cases distance alors il est normalisé de la manière suivante :

$$\bar{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\sum_{n=1}^{N} x(n)} \tag{1.19}$$

On a donc $\bar{\boldsymbol{x}} \in [0,1]$ et $\sum_{n=1}^{N} \bar{x}(n) = 1$. Le profil distance haute résolution normalisé peut donc être vu comme une distribution de probabilité discrète.

Une fois ces prétraitements réalisés, on peut extraire les attributs voulus des profils distance. Les premiers attributs extraits sont les moments centraux. Le moment central d'ordre k est calculé de la manière suivante :

$$m_k = \sum_{n=1}^{N} (n - n_0)^k \bar{x}(n) \tag{1.20}$$

où $n_0 = \sum_{n=1}^{N} n\bar{x}(n)$ est le moment origine d'ordre 1. Le premier vecteur d'attributs extraits des profils distance est donc de la forme $\boldsymbol{f}_1 = [m_2, m_3, \ldots, m_K]$. Ce vecteur est invariant en translation. La détermination du K optimal n'est pas évidente et les auteurs proposent de le déterminer empiriquement.

Une fois, ce premier vecteur extrait, on extrait un deuxième attribut des profils distance à savoir l'entropie. L'entropie d'un profil distance est défini de la manière suivante :

$$H = -\sum_{n=1}^{N} \bar{x}(n) log \bar{x}(n) \tag{1.21}$$

Le vecteur d'attributs final est donc $\boldsymbol{f}_2 = [m_2, m_3, \ldots, m_K, H]$. Il est de taille K et toujours invariant en translation. Si on note $\boldsymbol{F}$ la matrice contenant les M vecteurs attributs extraits des M profils distance de la base d'apprentissage, une dernière transformation est appliquée sur les f_{ij} de manière à ce que chaque vecteur d'attributs ait le même poids, ce qui selon les auteurs permet d'augmenter le taux de reconnaissance. Au final, on a :

$$\bar{f}_{ij} = \frac{f_{ij} - f_{i,min}}{f_{i,max} - f_{i,min}}, i = 1, 2, \ldots, M, j = 1, 2, \ldots, K \tag{1.22}$$

où $f_{i,min} = min(f_{ij})$ et $f_{i,max} = max(f_{ij}), j = 1, 2, \ldots, K$

Les résultats présentés montrent que la combinaison des deux vecteurs d'attributs permet d'augmenter les performances du classifieur KPPV par rapport aux performances obtenues en prenant les deux vecteurs d'attributs séparemment.

Un autre exemple d'article utilisant le classifieur des KPPV dans le contexte des profils distance haute résolution est l'article de Atrouz et al. en 2007 (Atrouz et al. 2007) où les auteurs proposent d'extraire des profils distance haute résolution d'images ISAR et d'appliquer un classifieur KPPV sur les profils distance haute résolution directement ou sur les attributs extraits d'une analyse en composantes principales. Les profils distance utilisés sont des profils distance synthétiques de 18 avions militaires. La base de données initiale a été séparée en deux parties, 75% des données sont utilisées pour la base d'apprentissage et 25% pour la base de test. L'algorithme est appliqué sur des profils distance en linéaire et alignés. Les auteurs ont comparé les performances en utilisant ces profils distance sans traitement et avec un prétraitement (fenêtrage de Hamming et compression non-linéaire Box-Cox). Les meilleures performances sont obtenues en appliquant l'algorithme de KPPV sur les profils distance haute résolution prétraités (99.7% de probabilité de bonne classification avec un prétraitement sur les profils distance HRD, 94.8% de probabilité de bonne classification sans prétraitement sur les profils distance HRD et 98% de probabilité de bonne classification avec prétraitement sur les composantes principales des profils distance HRD, 95.1% de probabilité de bonne classification sans prétraitement sur les composantes principales des profils distance HRD). Le nombre K de plus proches voisins a été fixé à $K = 1$ par les auteurs. Par contre, le type de métrique utilisée n'est pas précisé.

En 2011, Liu et al. (Liu et al. 2011) propose un algorithme modifié des KPPV, dit "Fuzzy KPPV", intégrant des concepts empruntés à la logique floue pour la reconnaissance de bateaux. Cet article s'appuie sur un article de 2006 du même auteur (Liu et al. 2006). L'intérêt de la méthode proposée par les auteurs réside dans la possibilité d'utiliser plusieurs types d'attributs différents pour réaliser la classification. Au lieu de travailler

sur la différence entre les attributs, on va en fait calculer une fonction d'appartenance associées à chacune de ces différences et prendre une décision à partir de ces fonctions d'appartenance ou plutôt à partir du degré d'appartenance (calculé à partir de la fonction d'appartenance) de l'attribut de test aux attributs d'apprentissage. La métrique utilisée par les auteurs est la distance de Manhattan (le nombre de plus proches voisins n'est pas précisé). Les résultats présentés montrent une amélioration conséquente des performances de reconnaissance par rapport à la méthode précédente présentée par l'auteur dans son article de 2006. Dans cet article, les concepts de la logique floue sont utilisés pour pouvoir prendre en compte plusieurs types d'attributs dans le processus de classification. Dans nos travaux de thèse, nous avons également repris les concepts de la logique floue (cf. chapitre 5) mais plutôt dans l'optique de prendre en compte dans le processus de classification, l'incertitude sur la mesure des profils distance.

L'enseignement que l'on peut tirer de ces articles est qu'il est important d'appliquer un prétraitement sur les profils distance avant d'utiliser l'algorithme des KPPV. Dans la suite de notre mémoire, nous reviendrons très souvent sur ces prétraitements. Les profils distance de notre base d'apprentissage subiront tous un fenêtrage de Hamming avant d'être utilisés dans nos algorithmes (cf. section 1.2.2). Nous utiliserons également nos profils distance à l'échelle logarithmique (correspondant à la compression non-linéaire Box-Cox mentionnée notamment dans l'article de Atrouz et Al.) de manière à rendre la distribution des données plus proche d'une distribution normale. En revanche dans la plupart des articles utilisant les KPPV, le problème du traitement du bruit sur les profils distance n'est pas abordé et cela car les profils distance utilisés sont des profils distance avec un RSB assez grand ($> 20dB$) et constant (même niveau de RSB entre profils distance de la base d'apprentissage et de la base de test). Mais dans le cas réel, on peut très bien avoir à traiter un profil distance de test avec un niveau de RSB différent du niveau de RSB des profils distance de la base d'apprentissage et dans ce cas là, il faut que l'algorithme de reconnaissance soit robuste aux différences de RSB. Une manière simple de faire, et que nous utiliserons pour les algorithmes présentés dans la suite du mémoire, est de seuiller les profils distance en estimant le niveau de bruit de ceux-ci. Nous reviendrons plus en détail sur cette étape lors de la présentation de chaque algorithme. Les paramètres de l'algorithme (nombre de plus proches voisins K ou métrique utilisée) ne sont généralement pas précisés dans les articles étudiés. Cela signifie sans doute qu'ils n'ont pas une influence majeure sur les performances de l'algorithme. Toutefois, nous étudierons l'influence de ces paramètres dans le chapitre du mémoire consacré à l'étude de l'algorithme des KPPV (cf. chapitre 3). La problématique de maîtrise du taux d'erreur n'est pas abordée dans ces articles car encore une fois les conditions de RSB semblent plutôt favorables. Cependant, il paraît très probable qu'avec des valeurs de RSB plus faibles, le taux d'erreur constaté avec ce type d'algorithmes augmenterait fortement. En effet, l'algorithme des KPPV ne présente pas de mécanismes permettant de prendre en compte l'incertitude sur la mesure des profils distance, le problème de maîtrise du taux d'erreur paraît donc difficile à

résoudre avec des algorithmes de ce type. Le chapitre 3 reviendra en détail sur les propriétés de l'algorithme des KPPV.

1.3.2 Classification paramétrique

Dans cette section, nous abordons les méthodes de classification de type paramétrique.

1.3.2.1 Méthodes bayésiennes

Lorsqu'il s'agit de méthodes de classification, on trouve très souvent dans la littérature et dans de nombreux domaines (Webb 2002), les méthodes dites *bayésiennes*. Un algorithme de type bayésien est proposé dans le chapitre 4 de ce mémoire. Les méthodes bayésiennes font partie des méthodes probabilistes. On les appelle bayésiennes car elles utilisent la règle de Bayes pour calculer la probabilité *a posteriori* à partir de laquelle la décision est prise. Il faut savoir qu'il existe d'autres types de méthode probabiliste utilisant d'autres types de règle de décision (règle du maximum de vraisemblance (MV), etc...) mais nous ne les détaillerons pas dans cette section.

On considère que l'on dispose d'une base d'apprentissage composée d'un certain nombre de profils distance issus de K classes différentes. On note k, la classe, on a donc $k = 1, \ldots, K$. Le but de ce type de méthode est d'estimer la probabilité *a posteriori* d'appartenir à une classe, c'est-à-dire $p(k|\boldsymbol{x})$, pour la classe k et un profil distance mesuré $\boldsymbol{x}$. En utilisant la règle de Bayes, on a

$$p(k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|k)p(k)}{\sum_k p(\boldsymbol{x}|k)p(k)} \tag{1.23}$$

avec $p(\boldsymbol{x}|k)$ la fonction de vraisemblance de la classe k et $p(k)$ la probabilité *a priori*.

Ces méthodes se résument généralement à estimer les paramètres des densités $p(\boldsymbol{x}|k)$ puis à calculer la probabilité *a posteriori* associée à chacune des classes. Les différents articles présents dans la littérature se différencient entre eux par le type de distributions utilisées pour modéliser les profils distance.

La thèse de Jacobs (Jacobs 1997) en 1997 est une étude complète sur la reconnaissance automatique de cibles à partir des profils distance haute résolution. Parmi d'autres méthodes de classification, Jacobs détaille un modèle de mélange de gaussiennes pour modéliser les profils distance. Les résultats présentés montrent que le modèle gaussien donne de meilleures performances par rapport à un modèle déterministe.

En 1999, Webb propose un modèle basé sur un mélange de distribution Gamma (Webb 2000). Il s'agit d'une application sur des profils distance de bateaux. La principale motivation de l'auteur dans cet article est de construire un classifieur robuste aux incertitudes sur l'amplitude des profils distance, la rotation et la translation de la cible. L'article présente donc différentes manières pour prendre en compte dans le modèle de mélange ces incertitudes. L'utilisation des distributions Gamma est justifiée par le

fait qu'elles englobent les distributions de Swerling 1,2,3 ou 4 qui sont des distributions souvent utilisées pour modéliser les propriétés statistiques de la SER d'objets complexes (Skolnik 2002).

L'article de Du et al. (Du and al. 2006), en 2006, détaille quant à lui un modèle de mélange basé sur deux types de distribution : les distributions Gamma et les distributions gaussiennes. Dans cet article, on considère que les cases distance d'un profil distance peuvent être catégorisées, selon le nombre de diffuseurs dans chaque cellule, en trois différents modèles statistiques. L'article développe donc un modèle statistique composé de deux formes paramétriques (gaussienne et Gamma), pour mieux modéliser les différents échos des différents types de cases distance. La catégorie des différentes cases distance est déterminée en utilisant un algorithme RPCL (Rival Penalized Competitive Learning) (Xu 1998). L'estimation des paramètres des lois Gamma du mélange est réalisée par la méthode du maximum de vraisemblance et l'estimation des lois gaussiennes par un algorithme Espérance-Maximisation (EM). Au final, les résultats présentés montrent que le modèle proposé offre de meilleures performances de reconnaissance mais également une plus grande robustesse par rapport aux deux modèles pris séparément à savoir le modèle utilisant des distributions Gamma et celui utilisant des distributions gaussiennes.

En 2011, Shi et al. (Shi et al. 2011) propose une méthode de reconnaissance statistique de profils distance haute résolution basée sur une analyse factorielle locale et un apprentissage bayésien "Ying-Yang Harmony" automatique. L'article s'appuie sur des études récentes comme celle de Du et al. (Du et al. 2008) qui propose une modélisation par analyse factorielle des profils distance haute résolution et sélection du modèle en deux phases. L'article va plus loin en modélisant la non gaussianité des profils distance haute résolution par une analyse factorielle locale (LFA pour Local Factor Analysis) que l'on appelle également MFA (pour Mixture of Factor Analysis). Puis les auteurs s'intéressent à l'étape de sélection du modèle qui d'après eux souffre, dans les études antérieures, d'une trop grande quantité de calculs et d'une mauvaise évaluation avec des profils distance de grande dimension. La méthode d'apprentissage bayésien automatique "Ying-Yang Harmony" proposée par l'auteur doit permettre de résoudre ces problèmes.

En 2011 également, Wang et Xie (Wang and Xie 2011) proposent de modéliser les profils distance par un modèle de mélange de distributions de Student. Les auteurs reprochent aux modèles de mélange de lois Gamma ou de mélange de lois Gamma et Gaussiennes d'être soit trop lourds en terme de charge de calcul ou trop compliqués. De plus, il a été montré (Mclachlan and Peel 2000) que les composantes gaussiennes d'un modèle de mélange présentent un manque de robustesse en présence de points aberrants. Les distributions de Student sont une solution pour résoudre ce problème. En effet, elles ont des queues de distributions plus épaisses que les distributions gaussiennes, ce qui diminue l'influence des points aberrants en réduisant leurs poids dans le calcul des paramètres. Pour estimer les paramètres, un algorithme de type EM (Expectation-

Maximization) donne des résultats assez médiocres lorsqu'une seule itération de l'algorithme est réalisée. Les auteurs ont préféré utiliser une version modifiée de l'algorithme EM à savoir un algorithme Fuzzy EM, pour estimer les paramètres du modèle de mélange de lois de Student. Les résultats présentés dans l'article semblent montrer qu'en plus d'obtenir de meilleurs résultats de reconnaissance que les autres types de modèles de mélange, cette méthode semble également plus robuste au bruit.

On trouve également un grand nombre d'articles qui, au lieu de modéliser les profils distance directement, vont plutôt modéliser des variables latentes issues de ces profils distance. Généralement, on cherche un espace où la limite de séparation des classes est plus nette que dans l'espace original et également, si possible, un espace de dimension plus faible. On a déjà vu par exemple avec l'article de Shi et al. (Shi et al. 2011) que ce ne sont plus les profils distance eux-même qui sont modélisés mais plutôt les facteurs issus d'une analyse factorielle ou d'une analyse en composantes principales probabiliste.

Application des méthodes bayésiennes au contexte de la thèse Pour résumer, l'approche que l'on retrouve dans la majorité des articles est une approche non supervisée qui consiste à modéliser la densité par un mélange de formes paramétriques dont les paramètres sont estimés dans la majorité des cas par un algorithme de type *Expectation-Maximization* (Redner and Walker 1984). Dans notre cas, l'estimation des paramètres peut se faire directement à partir de la base d'apprentissage car les classes des échantillons contenues dans cette base sont connues. L'algorithme *Expectation-Maximization* utilisé dans la majorité des articles pour estimer les paramètres ne nous sera donc pas utile car on se place dans un contexte supervisé. Egalement dans la plupart des articles, on ne modélise pas directement les profils distance mais on détermine plutôt des attributs à partir de ces profils distance qui sont ensuite modélisés. Cela permet de réduire la dimension du problème mais cela n'est pas forcément en adéquation avec l'objectif de la thèse qui est de définir des algorithmes se rapprochant d'algorithmes optimaux, c'est-à-dire compressant le moins possible l'information contenue dans les profils distance. Dans le cas où l'on cherche à modéliser les profils distance directement, il va se poser les problèmes du *fléau de la dimensionnalité*. Or cela n'apparaît pas forcément dans les articles présents dans la littérature. Nous étudierons plus en détail cet aspect dans le chapitre 4 du mémoire.

Parmi les modèles de mélanges utilisés, ceux utilisant des lois Gamma entraînent un coût de calcul très important et difficilement parallélisable, ce qui rend ce type de modèles difficile à utiliser. De même, le modèle proposé par Du en 2008, basé sur des distributions Gamma et gaussiennes, apparaît très complexe et difficilement applicable dans le cas réel. Les modèles basés sur des mélanges de distributions gaussiennes semblent être de bons candidats au vu des performances présentées dans les divers articles et de leurs simplicités d'utilisation. Les modèles de mélange de lois de Student doivent également être étudiés du fait de leur plus grande

robustesse, ce qui dans une problématique de maîtrise du taux d'erreur peut s'avérer très intéressant.

Dans tous les articles présentés ci-dessus, la règle de décision du maximum *a posteriori* (MAP) est utilisée, c'est-à-dire qu'on calcule la probabilité *a posteriori* associée à chaque classe et on affecte le profil distance sous test $\boldsymbol{x}$ à la classe ayant la probabilité *a posteriori* maximum. Dans notre contexte, où l'on cherche à maîtriser le taux d'erreur, cette règle n'est pas forcément bien adaptée. En effet, en utilisant cette règle, il y a la possibilité d'accepter pour une classe k un échantillon n'ayant par exemple que 40% de probabilité d'appartenir à cette classe. Nous avons donc réfléchi dans le chapitre 4 à une autre manière de prendre une décision, plus en adéquation avec l'objectif de maîtrise du taux d'erreur.

Une autre propriété des approches bayésiennes est qu'elles sont définies dans un monde fermé, c'est-à-dire que la classe de l'échantillon de test se trouve parmi les classes modélisées à l'aide de la base d'apprentissage et uniquement parmi ces classes. Avec ce type d'approche, il n'y a donc pas la liberté de dire que la classe de l'échantillon de test n'est pas parmi celles prises en compte dans le modèle. De plus, cette propriété se retrouve dès le calcul de la probabilité *a posteriori*. En effet, au moment de calculer la probabilité *a posteriori* d'appartenir à une classe k, celle-ci est normalisée par la somme des probabilités *a posteriori* associées à chaque classe du modèle de manière à avoir une valeur de probabilité entre 0 et 1. Les valeurs des probabilités *a posteriori* ne sont donc pas indépendantes les unes des autres. La décision prise pour la classe k n'est donc pas indépendante de celle prise pour la classe j. Jouer sur la règle de décision ne sera peut être donc pas suffisant pour maîtriser le taux d'erreur car la mesure même sur laquelle la décision est prise, est déjà normalisée pour pouvoir respecter les contraintes imposées par les méthodes bayésiennes. Nous étudierons cela dans le chapitre 4 du manuscrit.

1.3.2.2 Machines à vecteur de support ou Support Vector Machine (SVM)

Les SVM sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination/classification et de régression. Elles sont une généralisation des classifieurs linéaires. Les SVM standards prennent en entrée un ensemble de données et de prédictions et donnent en sortie pour chaque entrée, la classe à laquelle elle appartient parmi deux possibles. Il s'agit donc d'un classifieur non probabiliste binaire. Il existe des extensions des SVM aux cas multi-classes que l'on nomme multiclass SVM. L'approche utilisée dans ce cas consiste à réduire le problème multi-classes en une multitude de problème de classification binaire.

Dans le cas de la thèse, l'utilisation des SVM qui nous intéresse est celle permettant de résoudre les problèmes de classification. Il s'agit de décider à quelle classe appartient un échantillon $\boldsymbol{x}$. La résolution de ce problème passe par la construction d'une fonction h qui à un vecteur d'entrée $\boldsymbol{x}$ fait correspondre une sortie $\boldsymbol{y}$. Dans un premier temps, on se limite à un problème de classification à deux classes, c'est-à-dire $\boldsymbol{y} \in \{-1, 1\}$, le vecteur d'entrée $\boldsymbol{x}$ étant dans un espace X muni d'un produit scalaire.

Le cas simple est le cas d'une fonction discriminante linéaire, obtenue

par combinaison linéaire du vecteur d'entrée $\boldsymbol{x} = (x_1, \ldots, x_N)^T$, avec un vecteur de poids $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)$:

$$h(x) = \boldsymbol{\omega}^T \boldsymbol{x} + \omega_0 \tag{1.24}$$

Il est donc décidé que $\boldsymbol{x}$ est de classe 1 si $h(x) \geq 0$ et de classe -1 sinon. La frontière de décision $h(x) = 0$ est un hyperplan, appelé hyperplan séparateur ou séparatrice.

Le but d'un algorithme d'apprentissage supervisé est d'apprendre la fonction $h(x)$ par le biais d'un ensemble d'apprentissage :

$$(x_1, l_1), (x_1, l_1), \ldots, (x_p, l_p) \tag{1.25}$$

où les l_k sont les labels, p est la taille de l'ensemble d'apprentissage, N la dimension des vecteurs d'entrée. Si le problème est linéairement séparable, on doit alors avoir :

$$l_k h(x_k) \geq 0 \text{ avec } 1 \leq k \leq p$$

autrement dit

$$l_k(\omega^T x_k + \omega_0) \geq 0 \text{ avec } 1 \leq k \leq p$$

Par exemple dans une image, de manière assez simple, si dans le coin supérieur gauche on retrouve des pixels appartenant à la classe 1 et dans le coin inférieur droit des pixels appartenant à la classe 2, le problème sera linéairement séparable car on peut trouver une séparatrice qui sera la droite d'équation $y = x$. Par contre si les pixels de la classe 2 sont regroupés à l'intérieur d'un cercle et ceux de la classe 1 à l'extérieur, le problème n'est pas linéairement séparable. Il n'existe pas d'hyperplan séparateur dans ce cas. Nous reviendrons sur ce cas un peu plus tard mais attardons-nous tout d'abord sur le cas linéairement séparable.

En effet, même dans le cas simple exposé juste au-dessus, le choix de l'hyperplan n'est pas évident. Il existe une infinité d'hyperplans séparateurs. Pour résoudre ce problème, il a été montré (Vapnik 1982), qu'il existe un unique hyperplan optimal, défini comme l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur.

Il existe plusieurs méthodes pour maximiser cette marge. On peut résoudre ce problème en utilisant la méthode classique des multiplicateurs de Lagrange. Sans rentrer dans la formulation mathématique du problème, on peut dire que seuls les vecteurs supports, c'est-à-dire les échantillons les plus proches de l'hyperplan séparateur participent à la définition de l'hyperplan optimal. Ceci est donc efficace au niveau de la complexité et d'autre part, l'agrandissement de l'ensemble d'apprentissage a moins d'influence que dans un modèle de mélange gaussien par exemple, où tous les points participent à la solution.

Revenons à présent sur les cas non linéairement séparables. Dans cette situation, on utilise ce qu'on appelle le *Kernel trick*. L'idée est de reconsidérer le problème dans un espace de dimension supérieure, éventuellement de dimension infinie. Dans ce nouvel espace, il est alors probable qu'il existe une séparatrice linéaire. D'un point de vue formel, on applique au

vecteur d'entrée $\boldsymbol{x}$ une transformation non linéaire ϕ. L'espace d'arrivée $\phi(X)$ est appelé espace de redescription. Dans cet espace, on cherche alors l'hyperplan

$$h(x) = \boldsymbol{\omega}^T \phi(\boldsymbol{x}) + \omega_0 \tag{1.26}$$

qui vérifie $l_k h(x_k) > 0$, pour tous les points x_k de l'ensemble d'apprentissage, c'est-à-dire l'hyperplan séparateur dans l'espace de redescription. En utilisant la même procédure que dans le cas sans transformation, on aboutit à un problème d'optimisation équivalent au cas sans transformation. Le problème est que cela implique l'existence d'un produit scalaire entre vecteurs dans l'espace de redescription, de dimension élevée, ce qui est couteux en terme de calculs. C'est à ce niveau là qu'intervient l'astuce connue sous le nom de *Kernel trick*, qui consiste à utiliser une fonction noyau de la forme

$$K(x_i, x_j) = \phi(x_i)^T.\phi(x_j) \tag{1.27}$$

L'intérêt de la fonction noyau est que le calcul se fait dans l'espace d'origine, ce qui est beaucoup moins couteux qu'un produit scalaire en grande dimension. De plus, la transformation ϕ n'a pas besoin d'être connue explicitement, seule la fonction noyau intervient dans les calculs. Les noyaux usuels employés avec les SVM sont les noyaux polynomiaux ou gaussiens.

En 2005, Chen et al. (Chen et al. 2005) proposent également une méthode basée sur les SVM pour faire de la reconnaissance de profils distance haute résolution. Dans un premier temps, les auteurs proposent d'utiliser une ACP locale pour faire de la reconnaissance de profils distance. On parle d'ACP locale car les auteurs proposent de diviser la base d'apprentissage en plusieurs clusters et ensuite d'effectuer une ACP pour chaque cluster. Pour faire le clustering, l'article propose d'utiliser un algorithme de quantification vectorielle non supervisée, à savoir l'algorithme LBG (Linde-Buzo-Gray) (Linde et al. 1980). Il s'agit d'un algorithme LBG un peu adapté à la spécificité des profils distance. En effet, l'algorithme LBG traditionnel est basé sur la minimisation du critère distance euclidienne. Or, avant de pouvoir calculer la distance euclidienne entre deux profils distance, ceux-ci doivent être alignés. Au lieu d'utiliser la distance euclidienne, les auteurs définissent une nouvelle métrique qui calcule l'ensemble des distances entre deux profils distance dont un est décalé cycliquement et qui retient la distance minimale. Après avoir défini comment extraire les attributs des profils distance avec une ACP localisée, les auteurs proposent une autre méthode d'extraction d'attributs basée sur un noyau ACP. La méthode d'extraction par ACP permet d'extraire les informations concernants les structures linéaires de la base de données. Les méthodes à noyau et notamment le noyau ACP (Kocsor and Toth 2004) permettent d'extraire des structures non-linéaires de la base de données. Comme pour l'ACP localisée, le noyau ACP est défini de façon à prendre en compte le décalage entre les profils distance. La métrique définie pour l'ACP localisée est réutilisée pour définir le noyau ACP. Au final, les résultats montrent que ces méthodes permettent d'améliorer

les performances de reconnaissance tout en réduisant la dimension du système et en traitant l'alignement des profils distance. L'ACP localisée est plus performante que l'ACP simple en terme de performance de reconnaissance par contre le temps d'exécution est plus important. En revanche, le noyau ACP ou kernel ACP (KACP) obtient les meilleures performances en terme de reconnaissance et en terme de temps d'exécution. La même année, Liu et al (Liu et al. 2005a) étendent ces travaux en définissant un noyau ACI (KACI) pour un classifieur SVM qui permet d'améliorer encore légèrement les performances.

En 2008, Li et al. (Li et al. 2008) proposent un noyau gaussien amélioré pour un classifieur SVM et une nouvelle méthode pour la sélection des paramètres du classifieur SVM basée sur une combinaison des méthodes "Leave-One-Out" et "One-Validation". Les résultats présentés montrent que l'algorithme SVM amélioré permet d'améliorer les performances de reconnaissance d'un algorithme SVM classique et d'être plus robuste à la taille de la base de données, à l'angle d'aspect et au bruit.

Enfin en 2008, Kent et al. (Kent et al. 2008) comparent les performances d'un classifieur SVM avec deux méthodes de classification statistique, une basée sur le maximum de vraisemblance et l'autre basée sur la vraisemblance linéaire de Fisher, le tout appliqué à des profils distance haute résolution. Le classifieur SVM étudié utilise un noyau à base radiale (dont la valeur dépend uniquement de la distance à l'origine). Les résultats présentés mettent en évidence que le classifieur SVM permet d'obtenir de meilleures performances de classification que les deux autres classifieurs mais également d'être plus robuste au bruit.

Le problème qui va se poser avec ce type d'algorithmes est le même qu'avec les algorithmes de type KPPV par exemple. Au contraire, des méthodes de type logique floue ou bayésiennes, les SVM ne permettent pas de prendre en compte les incertitudes sur la mesure des profils distance. Si les profils distance mesurés ne sont pas de bonnes qualités (c'est-à-dire que la valeur de la SER varie beaucoup dans chaque case distance) et que cela n'est pas pris en compte, à un moment donné, dans l'algorithme de reconnaissance, le taux d'erreur risque de devenir important et on s'expose donc à des erreurs de classification qui dans le contexte NCTR, où des vies humaines sont en jeu, ne sont pas acceptables.

1.3.2.3 Reconnaissance de cibles par logique floue

On trouve dans la littérature quelques méthodes de reconnaissance de cibles basées sur le principe de la logique floue. Nous détaillerons dans le chapitre 5 de ce mémoire, deux algorithmes, mis au point dans le cadre de la thèse, basés sur les principes de la logique floue. La logique floue est une technique qui s'appuie sur la théorie mathématique des ensembles flous. Cette théorie, introduite par Zadeh (Zadeh 1965), est une extension de la théorie des ensembles classiques pour la prise en compte d'ensembles définis de façon imprécise. C'est une théorie formelle et mathématique dans le sens où Zadeh, en partant du concept de fonction

d'appartenance pour modéliser la définition d'un sous-ensemble d'un univers donné, a élaboré un modèle complet de propriétés et de définitions formelles. Il a aussi montré que cette théorie des sous-ensembles flous se réduit effectivement à la théorie des sous-ensembles classiques dans le cas où les fonctions d'appartenance considérées prennent des valeurs binaires (0,1).
Les applications de reconnaissance de cibles basées sur la logique floue s'appuient sur deux fonctions de base :

- Fonction d'appartenance $M(\theta)$: cette fonction est utilisée pour décrire comment une classe peut être représentée par rapport à un paramètre θ. Elle traduit notre connaissance *a priori* sur les valeurs possibles que peut prendre ce paramètre pour une classe donnée.

- Densité de possibilité $D(\theta)$: elle décrit comment la valeur réelle d'un paramètre peut être distribuée, étant donné la mesure de ce paramètre et l'estimation de sa précision.

En combinant les deux fonctions ci-dessus, deux valeurs sont calculées pour chaque classe :

- la possibilité : cette valeur décrit le degré d'intersection entre la mesure et les données de référence de chaque classe. Soit θ un paramètre. Connaissant la fonction d'appartenance et la densité de possibilité associée à ce paramètre, on en déduit la valeur de la possibilité de la manière suivante :

$$P = \max_{\theta}\{\min[M(\theta), D(\theta)]\} \tag{1.28}$$

- la nécessité : cette valeur décrit le degré d'inclusion de D dans M.

$$N = \min_{\theta}\{\max[M(\theta), 1 - D(\theta)]\} \tag{1.29}$$

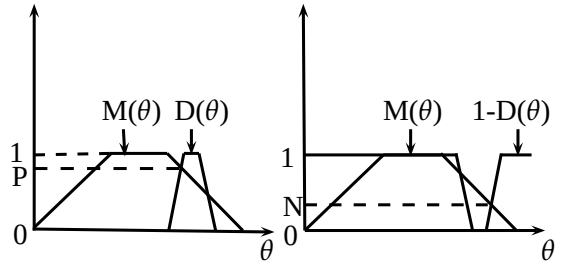


FIG. 1.9 : Calcul de la possibilité (P) et de la nécessité (N)

Les possibilités et nécessités associées à chaque paramètre sont donc calculées et une décision est prise à partir de ces grandeurs.

En 1998, Moruzzis et Colin (Moruzzis and Colin 1998) propose une méthode de classification de cibles basée sur les principes de la logique

floue pour les radars conventionnels et multifonctions. Les auteurs présentent un exemple d'algorithme pour la classification de cibles de type hélicoptère, chasseurs, missiles ou drônes. Les résultats montrent que la prise en compte des erreurs de mesure via la densité de possibilité permet de limiter les fausses décisions. En utilisant de la fusion temporelle, les auteurs montrent que les performances sont améliorées.

En 2004, Moruzzis et al. (Moruzzis et al. 2004) présentent dans un article les résultats du programme de recherche MILORD qui avait pour but d'évaluer différentes techniques NCTR dans un environnement réel et de valider l'intérêt de la modélisation de cibles. Cet article résume les avantages des différentes techniques NCTR et revient également sur l'intérêt de la logique floue pour prendre en compte les erreurs de mesure et fusionner les décisions prises pour l'ensemble des échantillons d'un même attribut ou d'attributs de natures différentes.

L'avantage de la logique floue est qu'elle est bien adaptée aux données incertaines, les erreurs de mesures étant explicitement prises en compte. De plus, les mêmes méthodes peuvent être utilisées pour fusionner des paramètres de natures différentes tels que des paramètres continus ou discrets, provenant de différentes sources (fusion spatiale) et provenant d'instants différents (fusion temporelle). La fusion spatiale peut être intéressante car elle permet d'utiliser des attributs provenant de plusieurs sources pour prendre une décision mais cet aspect de la logique floue ne sera pas exploité dans cette thèse.

1.4 Espace de représentation

Dans la section 1.3, nous avons présenté certaines méthodes de classification de profils distance que l'on trouve dans la littérature. Certains articles avant de passer à l'étape de classification proposent d'appliquer une transformation sur les profils distance de manière à les représenter dans un autre espace, généralement de plus faible dimension et dans lequel les composantes ont des propriétés intéressantes pouvant faciliter l'étape de classification. Dans cette section, nous étudions les transformations que l'on peut trouver dans la littérature concernant les profils distance. Nous revenons également sur certains attributs de classification ou méthodes de classification lorsque ceux-ci ou celles-ci n'ont pas été mentionnés dans la section 1.3. Dans le paragraphe 1.4.1, nous étudions les techniques NCTR existantes dans le domaine fréquentiel puis nous nous intéressons à quelques travaux réalisés dans le domaine des ondelettes (cf. paragraphe 1.4.2) mais également à des travaux utilisant les représentations parcimonieuses (cf. paragraphe 1.4.3).

1.4.1 Profils distance haute résolution dans le domaine fréquentiel

Comme on l'a mentionné dans la section 1.2.5, certaines études se sont penchées sur les techniques NCTR dans le domaine fréquentiel. L'article de Wong (Wong 2004) détaille les différents prétraitements pouvant être réalisés sur les profils distance haute résolution :

- Alignement des profils distance entre eux ;
- Compensation de la migration de la cible ;
- Détection de la tête et de la queue de la cible.

Il revient sur chacun de ces prétraitements et montre en quoi l'utilisation du module de la transformée de Fourier du profil distance permet d'éviter de tels prétraitements. L'auteur revient également sur la perte de l'information de phase qu'il considère comme négligeable au regard de toutes les simplifications que permet l'utilisation de ce type de signaux par rapport aux profils distance.

L'article de Zyweck et Bogner (Zyweck and Bogner 1996) utilise également ce type de transformation pour s'affranchir des prétraitements nécessaires lorsqu'on utilise les profils distance.

En 2011, Wang et al. (Wang et al. 2011), proposent une méthode de reconnaissance de profils distance haute résolution dans le domaine fréquenciel basée sur un modèle autorégressif. L'article propose donc un modèle pour caractériser l'amplitude du spectre fréquentiel des profils distance haute résolution et d'en extraire les coefficients autorégressifs (AR) et les coefficients de corrélation partielle (PARCOR) pour les utiliser comme des attributs discriminatoires. Ces attributs ont l'avantage d'être invariants à la phase initiale, en translation et aux changements d'échelle. Le papier reprend le principe d'apprentissage bayésien "Ying-Yang Harmony" déjà vu dans l'article (Shi et al. 2011) pour déterminer automatiquement la répartition des profils distance selon leur angle d'aspect. Le but est de découper au mieux la base d'apprentissage en "frame", chacun des profils distance appartenant à une frame étant considéré comme suivant une même distribution. Le nombre de frames correspond donc au nombre de distributions du modèle de mélange utilisé. Bien sur, on raisonne avec les profils distance d'une même classe. La même opération est réalisée pour chacune des classes séparemment. Les résultats présentés montrent que les performances de reconnaissance sont un peu inférieures à celles obtenues dans l'article de Du et al. (Du et al. 2008) qui utilisent une modélisation par analyse factorielle. Par contre, les attributs extraits du modèle autorégressifs sont ceux ayant la dimension la plus faible et le temps de traitement pour la reconnaissance d'un profil distance de test est beaucoup plus faible que le temps nécessaire avec la méthode de l'article (Du et al. 2008).

L'utilisation des profils distance haute résolution dans le domaine fréquentiel est certes très pratique car cela permet de s'affranchir de certains prétraitements qui s'avèrent obligatoires dans le domaine temporel. Néanmoins, cela entraîne inévitablement une perte d'information qui pourrait s'avérer discriminante. Ce deuxième point va à l'encontre de l'esprit de la thèse où on cherche vraiment à s'approcher d'algorithmes optimaux pour lesquels l'information est comprimée au minimum.

1.4.2 Ondelettes

Il s'agit d'utiliser les ondelettes pour décomposer les profils distance. La transformée en ondelette reprend le formalisme de la transformée de Fourier. Cependant la transformée de Fourier entraîne une perte de tous les aspects temporels du signal. La transformation en ondelette permet d'analyser le signal à la fois en temps et en fréquence. Le principe est de représenter un signal par la somme pondérée d'une ondelette translatée ou dilatée. On parle généralement de famille d'ondelettes, constituée d'une ondelette mère Φ et de l'ensemble de ses images $\phi_{s,\tau}$

$$\forall t \in \mathcal{R}, \phi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Phi\left(\frac{t-\tau}{s}\right) \tag{1.30}$$

On définit la transformée en ondelette discrète de la manière suivante sachant que X est la matrice des profils distance et $\tilde{X}$, celle des coefficients d'ondelettes.

$$\tilde{X}(t) = \sum_{m\in\mathcal{Z}}\sum_{n\in\mathcal{Z}}\langle X, \phi_{m,n}\rangle.\phi_{m,n}(t) \tag{1.31}$$

Une application aux profils distance est détaillé dans l'article de Liu et al. de 2005 (Liu et al. 2005b). Les auteurs proposent une méthode de reconnaissance de profils distance haute résolution basée sur les transformations en paquet d'ondelettes et une décomposition en sous-bandes. On cherche donc à classifier un profil distance. La première étape consiste à décomposer ce profil distance en plusieurs sous-bandes en utilisant un paquet d'ondelettes basé sur les ondelettes de Daubechies. Pour chaque sous-bande, un classifieur est appliqué. La décision finale est la fusion de l'ensemble des décisions prises par chaque classifieur. Plusieurs règles de décision sont proposées par les auteurs : la règle du produit et de la somme (Kittler et al. 1998) ou la méthode d'intégration des croyances (Chen and Tang 2004). Les résultats montrent que les performances sont meilleures lorsque l'on décompose en sous-bandes en utilisant la règle de fusion du produit ou d'intégration des croyances. En utilisant la règle de fusion de la somme, les performances sont moins bonnes qu'en mono-bande.

Brousseau en 2012 (Brousseau 2012) a également utilisé des structures hiérarchiques d'ondelettes de type "arbre" pour représenter une base de données de profils distance synthétiques dans le contexte NCTR. En effet, la taille des bases de données de profils distance devient de plus en plus grande du fait de l'augmentation de la résolution des radars récents. Ces structures permettent de réduire la taille de ces bases de données de manière efficace, ce qui permet de réduire les ressources mémoires nécessaires mais également le temps d'exécution du processus de reconnaissance. Brousseau a utilisé un algorithme de type KPPV (en utilisant comme métrique la distance euclidienne) pour faire la reconnaissance des cibles et tester l'efficacité de la compression de la base de données. Les résultats montrent que le coût de calcul a été divisé par 13 avec une dégradation de RSB pour atteindre une probabilité de fausse classification inférieure à 1% de $8dB$.

L'utilisation des ondelettes dans le domaine NCTR peut être très intéressante notamment lorsqu'il s'agit de traiter de problématiques telles que la réduction des bases de données. Néanmoins, l'objectif de la thèse est de se focaliser sur l'étude de traitements optimaux qui évitent au maximum de compresser l'information. En tenant compte de ces considérations, l'utilisation des ondelettes telle qu'elle est présentée dans les articles présents dans la littérature n'entre pas en adéquation avec les objectifs de la thèse.

1.4.3 Représentations parcimonieuses

Les représentations parcimonieuses ont reçu une attention croissante dans de nombreux domaines ces dernières années. La théorie des représentations parcimonieuses considère que le signal est un mélange d'un large éventail de formes d'onde avec différentes propriétés fréquentielles, temps-échelle ou temps-fréquence. Les méthodes conventionnelles décomposent le signal en un seul type de formes d'onde, les représentations parcimonieuses analysent le signal sur la base d'un dictionnaire redondant, constitué d'un nombre important de formes d'onde différentes, appelées *atomes*. Comparées aux méthodes conventionnelles, les structures locales du signal peuvent être individuellement adaptées aux *atomes*. On peut résumer la théorie des représentations parcimonieuses de la manière suivante. Soit x un profil distance. On cherche à exprimer $x = (x_1, ..., x_N)$ de la manière suivante :

$$\underline{\boldsymbol{x}} = A\tilde{X} = \begin{pmatrix} A'_1 & A'_2 & \dots & A'_M \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_M \end{pmatrix} = \sum_{i=1}^{M} A'_i \tilde{x}_i \tag{1.32}$$

où la matrice A de taille $N_s \times M$ est le dictionnaire redondant. Chaque A'_i de taille $N_s < M$ est appelé *atomes*. $\tilde{X}$ est le vecteur des coefficients parcimonieux. Comme $N_s < M$, le problème est sous déterminé. Il existe donc une infinité de solutions. Parmi toutes ces solutions, on considère que celle où $\tilde{X}$ possède un nombre faible de coefficients différents de zéro est la meilleure. Les deux principales tâches pour construire ce type de représentation sont la construction du dictionnaire et la résolution du problème d'optimisation. Deux solutions principales existent pour construire ce dictionnaire. La première consiste en une collection de g' :

$$g' = \frac{1}{\sqrt{s}} g\left(\frac{t-u}{s}\right) e^{-\pi \nu t^2} \tag{1.33}$$

où les g' sont des transformations de la fonction de base g et s,u et ν sont respectivement des coefficients de dilatation, translation et de modulation de fréquence. Les différents g' produits ont des propriétés fréquentielles, temps-échelle et temps-fréquence variées. Avec ce type de collection, le dictionnaire peut être aussi redondant qu'on le souhaite. Cependant dans la pratique, il peut parfois être difficile de décider le bon nombre d'atomes composants le dictionnaire. De plus, comme chaque atome provient d'une même fonction de base, l'adaptabilité de la représentation des composantes du signal est relativement limitée. La deuxième solution consiste à construire le dictionnaire à partir d'une base d'apprentissage. Dans ce

cas le nombre d'atomes peut être décidé plus facilement. La deuxième étape consiste à trouver une solution optimale (nombre de coefficients différents de zéro faible) à l'équation 1.32. Dans la littérature, on trouve plusieurs algorithmes permettant de résoudre ce problème très complexe. Nous citerons les trois principaux, l'algorithme *Matching pursuit* (Mallat and Zhang 1993), l'algorithme *Basis pursuit* (Chen et al. 1999) et enfin la méthode *LASSO* (Donoho and Tsaig 2008).

En 2009, Nuo et al. (Nuo and al. 2010) proposent une nouvelle méthode de reconnaisssance pour des cibles terrestres. Cette méthode est basée sur les représentations parcimonieuses et l'utilisation des coefficients parcimonieux extraits des profils distance haute résolution comme attributs de classification. Les auteurs construisent le dictionnaire à partir d'une base de profils distance d'apprentissage. Le calcul des coefficients parcimonieux est réalisé en utilisant la méthode *LASSO*. L'article compare les performances obtenues en exploitant un algorithme de type "template based" en utilisant l'amplitude des profils distance ou les coefficients parcimonieux extraits. Les résultats montrent que l'utilisation des coefficients parcimonieux augmente les performances de reconnaissance. De plus, la méthode semble plus robuste notamment à l'augmentation du nombre de cibles à détecter.

Comme pour les ondelettes, les représensations parcimonieuses ne sont pas forcément en adéquation avec la philosophie de la thèse. En effet, elles ont inévitablement tendance à réduire la dimension du problème et on s'éloigne donc avec ce genre d'approche de traitements de type "force brute" que l'on souhaite tester au cours de la thèse.

1.5 Conclusion du chapitre

Cet état de l'art nous a permis d'avoir une vision globale du domaine de la reconnaissance de cibles non coopératives. Une description des différents attributs utilisés pour faire la reconnaissance a été réalisée et, compte tenu des spécificités de chacun, les profils distance haute résolution semblent être de bons candidats pour répondre aux objectifs définis par la thèse. Les différents articles nous ont permis également d'identifier les difficultés que l'on peut rencontrer avec ce type d'attributs (alignement, seuillage, etc...) et d'avoir une idée des solutions existantes afin de les contourner. Nous nous sommes ensuite focalisés sur les méthodes de classification qui sont utilisées dans le domaine de la reconnaissance de cibles non coopératives. De nombreuses approches ont été étudiées mais, au final, on se rend compte que la thématique de maîtrise du taux d'erreur n'est pratiquement jamais abordée. Les solutions pour faire face à cette problématique avec des approches de type KPPV ou SVM semblent assez limitées, en revanche les approches bayésiennes et surtout logique floue semblent fournir des mécanismes permettant de traiter ce type de contraintes. En tenant compte de ces conclusions, nous nous sommes orientés durant nos travaux de thèse vers trois types d'approches :

- L'approche de type KPPV
- L'approche bayésienne

– L'approche logique floue

La figure 1.10 résume dans un graphe les différentes notions vues dans cet état de l'art. Ce graphe permet d'avoir une vision globale de ce qui se fait dans le domaine NCTR et de situer les axes de travail de la thèse (en rouge) dans ce vaste domaine de recherche que constitue la reconnaissance de cibles non coopératives. Nous avons fait également apparaître au niveau des transformations sur le profil distance, deux méthodes (ACP et t-SNE pour t-Distributed Stochastic Neighbor Embedding) que nous avons utilisées dans la thèse pour visualiser les données et qui seront détaillées un peu plus tard dans le mémoire, respectivement dans les sections 2.3.3 et 4.6.

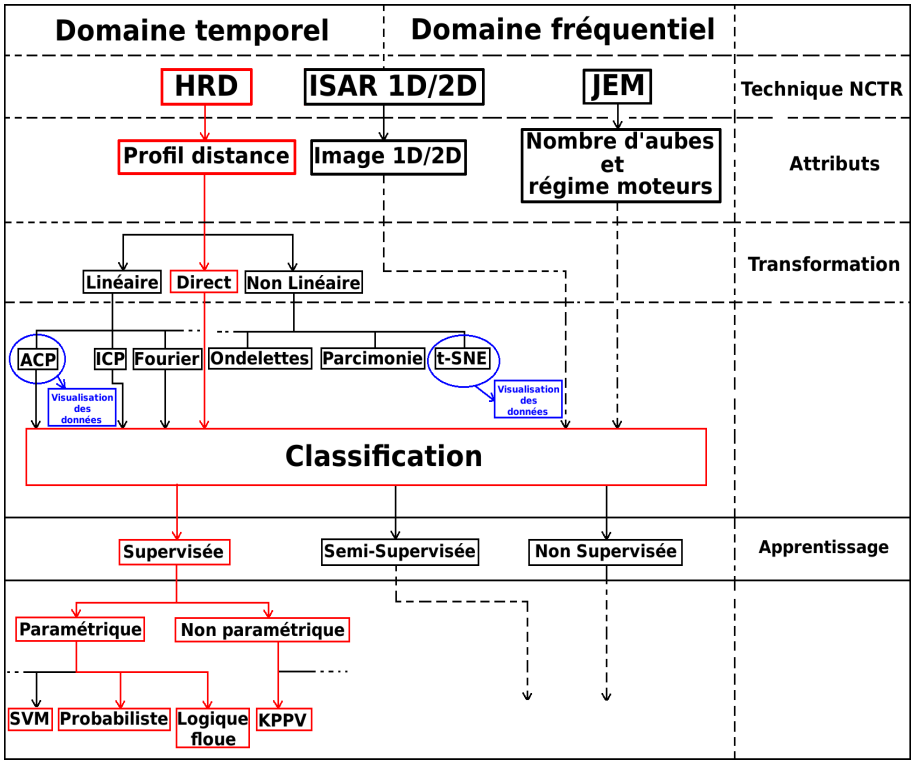


Fig. 1.10 : Synthèse de l'état de l'art

Dans la suite de ce mémoire, nous détaillons dans un premier temps les données utilisées au cours de la thèse pour évaluer nos algorithmes puis nous proposons des algorithmes basés sur ces trois approches et nous les étudions en les confrontant aux objectifs de la thèse.

Présentation des données et caractérisation statistique

2

Sommaire

2.1 Base de données 41
2.1.1 Données synthétiques 41
2.1.2 Données réelles 43
2.2 Visualisation simple de la base de données synthétiques 43
2.3 Etude statistique des matrices de covariance 46
2.3.1 Inertie des matrices de covariance 47
2.3.2 Volume des classes 50
2.3.3 Représentation graphique des données 53
2.4 Conclusion du chapitre 59

Le but de ce chapitre est de présenter les données utilisées au cours de la thèse pour tester nos algorithmes. Dans la section 2.1, nous présentons deux types de données : des données synthétiques et des données réelles. Nous nous attardons plus précisemment sur la base de données synthétiques (cf. chapitre 2.1.1) en présentant la base d'apprentissage et la base de test puis en s'intéressant dans la section 2.2 à une première méthode simple pour visualiser les données et dans la section 2.3 aux propriétés statistiques de ces deux bases. En s'appuyant sur cette étude statistique, nous proposons dans le paragraphe 2.3.3 une représentation graphique 2D des bases d'apprentissage et de test.

2.1 Base de données

Durant la thèse, nous avons utilisé deux types de données : des données synthétiques et des données réelles. Nous revenons dans les paragraphes suivants sur les caractéristiques de ces bases de données.

2.1.1 Données synthétiques

Elles sont découpées en deux bases : une base d'apprentissage et une base de test. Chacune de ces deux bases contient trois types de chasseurs

différents. Par commodité, on employera dans la suite du mémoire le mot "classe" pour parler des différents types de chasseurs.

Base d'apprentissage

La base d'apprentissage contient $N_A = 1024$ profils distance répartis équitablement dans $K = 3$ classes différentes ($N_A^k = 341$ avec $k = 1, \ldots, K$). Un profil distance artificiel a été ajouté de manière à avoir $N_A = 1024$, ce qui simplifiera l'implémentation sur GPU présentée dans le chapitre 6. Les profils distance de la base d'apprentissage ont été obtenus à partir de modélisation CAO des cibles sur lesquelles ont été appliquées des techniques de calcul de SER. Pour chaque configuration (fréquence, angle de site, azimut), les valeurs complexes de SER sont calculées. La base d'apprentissage est en fait restreinte en limitant la variation de l'angle d'azimut et de site autour de l'azimut α_t et du site θ_t des profils distance de la base de test, qui ont tous le même azimut et site. On sélectionne donc les profils distance d'apprentissage à partir de l'angle d'azimut et de site estimés et de leurs précisions (σ_α et σ_θ), c'est-à-dire dans l'intervalle $[\alpha_t - 3\sigma_\alpha; \alpha_t + 3\sigma_\alpha]$ pour l'azimut et $[\theta_t - 3\sigma_\theta; \theta_t + 3\sigma_\theta]$ pour le site. Chaque profil distance d'apprentissage de la même classe est recalé en distance. L'étape de recalage consiste à définir un profil d'apprentissage dans chaque classe comme référence et de recaler tous les autres par rapport à ce profil distance de référence. Pour les recaler, on calcule la distance euclidienne entre le profil distance à recaler et le profil distance de référence et cela en décalant le profil distance de quelques cases distance à chaque fois. La distance minimale sur l'ensemble de ces décalages permet de déterminer le recalage à appliquer. Il s'agit donc d'un alignement de type relatif (cf. section 1.2.5).

Base de test

La base de test est constituée de $N_T = 150$ profils distance également répartis équitablement dans 3 classes différentes ($N_T^K = 50$ avec $k = 1 \ldots K$). Les profils distance de la base de test ont tous le même angle d'aspect (α_t, θ_t). Ils ont été obtenus par simulation à partir des profils de la base d'apprentissage en ajoutant un bruit de type gaussien représentatif de la chaîne de réception radar. Contrairement à la base d'apprentissage, les profils distance de test ne sont pas recalés en distance et cela se rapproche au maximum d'une situation opérationelle réelle. En effet, quand un profil distance est mesuré par un radar, il faut le recaler en distance par rapport à la base d'apprentissage et cette étape de recalage peut être affectée par un RSB plus ou moins élevé.

Niveaux de RSB

Un des enjeux des algorithmes développés dans le cadre de la thèse consiste à contrôler le taux d'erreur quel que soit le RSB du profil distance mesuré. Pour les tester, un bruit blanc gaussien a été rajouté aux signaux IQ initiaux afin de modifier le RSB des profils distance de test. Afin de pouvoir superposer base d'apprentissage et base de test, nous avons également rajouté du bruit sur les signaux de la base d'apprentissage afin d'avoir un RSB équivalent à celui observé sur les signaux de

la base de test. En revanche, pour tester les performances de nos algorithmes (cf. chapitres 3, 4 et 5), seule la base de test est bruitée, la base d'apprentissage n'étant pas bruitée. Pour ce chapitre, il existe finalement trois configurations pour la base d'apprentissage et de test :

- Une première configuration où le RSB moyen des profils distance est égal à $30dB$.
- Une deuxième configuration où le RSB moyen des profils distance est égal à $20dB$.
- Une troisième configuration où le RSB moyen des profils distance est égal à $15dB$.

2.1.2 Données réelles

Il s'agit de données issues d'une campagne d'enregistrement en environnement réel sur trois types de chasseurs.

2.2 Visualisation simple de la base de données synthétiques

Les figures 2.1, 2.2 et 2.3 représentent la superposition des profils distance de la base d'apprentissage et de la base de test pour chacune des trois classes et pour les trois niveaux de RSB, dans le cas où les profils distance ne sont pas seuillés. Les profils distance de la base de test n'étant pas recalés entre eux, on a appliqué une processus d'alignement relatif de ces profils distance avant de les superposer. On a également représenté en noir, les profils distance moyens de chaque classe. Cette simple superposition permet d'avoir un premier aperçu de la forme des signaux suivant les classes.

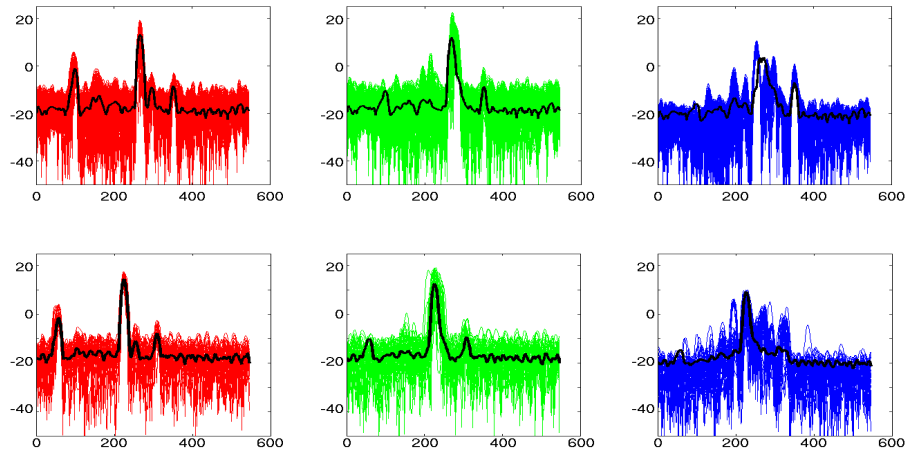


Fig. 2.1 : Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 30dB$

En observant la superposition des profils d'apprentissage, on arrive plutôt bien à distinguer la forme des profils distance des différentes classes. Les profils distance moyens des différentes classes sont plutôt bien différenciables même si on voit que ceux des classes 1 et 2 sont tout de

même relativement proches. On remarque également que les profils distance de test et d'apprentissage sont relativement proche entre eux.

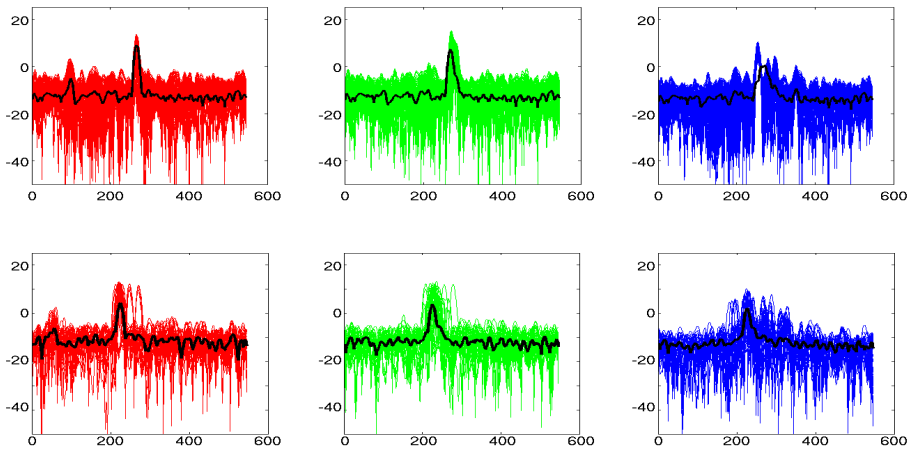


FIG. 2.2 : Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 20dB$

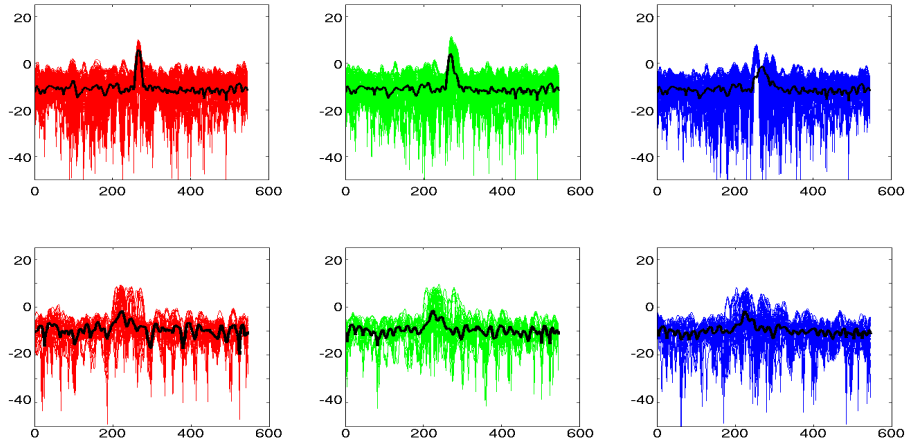


FIG. 2.3 : Superposition des profils distance de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 15dB$

Plus le RSB devient faible et plus les classes 1 et 2 deviennent difficiles à distinguer. La classe 3 reste toujours un peu plus facilement identifiable par rapport aux deux autres. On constate également que plus le RSB est faible et moins l'étape de recalage est précise car les échantillons de bruit prennent une place de plus en plus importante dans le calcul des distances.

Les figures 2.4, 2.5 et 2.6 représentent également une superposition des profils distance de chaque classe mais cette fois-ci dans le cas où les profils distance sont seuillés. Pour cette superposition et pour la caractérisation statistique qui suit, on applique un même seuil en amplitude (pour filter les échantillons de bruit) à tous les profils distance et cela dans un souci de visibilité des figures de superposition. Cela permet également d'obtenir une meilleure caractérisation statistique des différentes classes. Il est

également possible de faire un fenêtrage temporel pour seuiller les profils distance mais ces deux types de seuillage donnent des résultats à peu près semblables (Boulay 2010). Par contre, on verra dans la suite du mémoire que, pour certains algorithmes, ce seuil pourra varier suivant les profils distance. La valeur du seuil appliqué sur les profils distance dans ce chapitre correspond à la valeur moyenne des seuils calculés pour chaque profil distance. La valeur du seuil pour chaque profil distance est déterminée à partir de la puissance de bruit calculée sur ces profils distance. En effet, sur un profil distance, la compression d'impulsion permet de faire ressortir la signature électromagnétique de la cible (localisée dans une portion de la fenêtre d'acquisition) des échantillons de bruit. En supposant que la fenêtre d'acquisition est centrée sur la cible, les premiers et derniers échantillons du profil distance correspondent alors aux échantillons de bruit, à partir desquels la puissance de bruit peut être estimée.

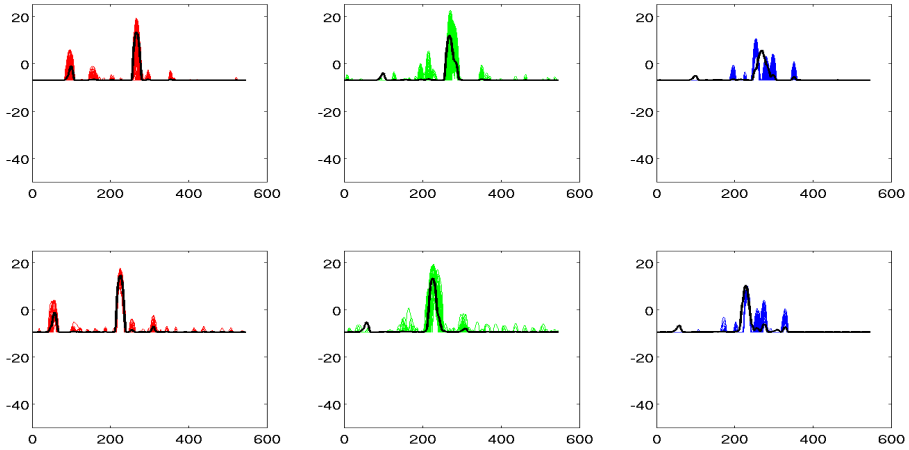


FIG. 2.4 : Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 30dB$

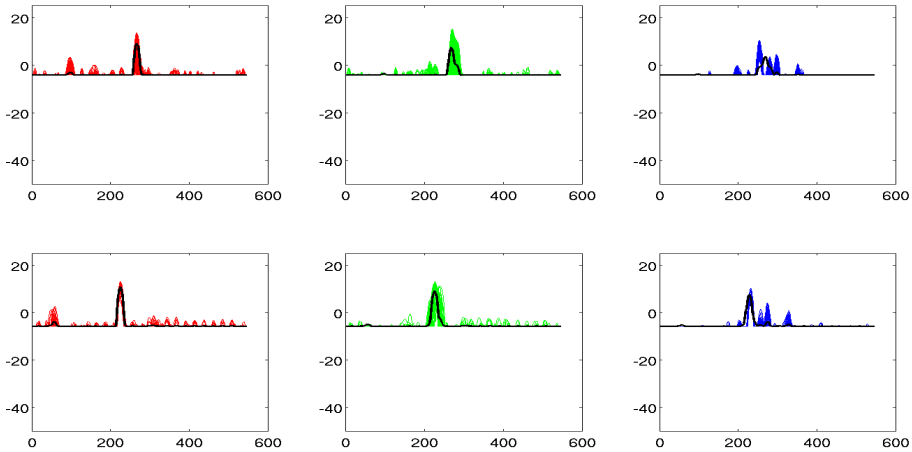


FIG. 2.5 : Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 20dB$

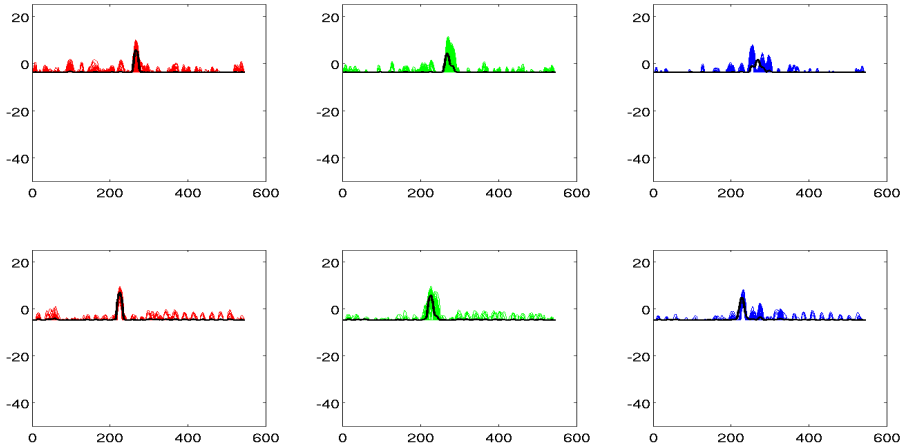


FIG. 2.6 : Superposition des profils distance seuillés de chaque classe de la base d'apprentissage en haut et de la base de test en bas pour un $RSB = 15dB$

Sur les figures 2.4, 2.5, 2.6, on voit bien que le seuillage permet d'éliminer un maximum d'échantillons de bruit pour pouvoir faire la reconnaissance uniquement sur le signal utile. Cette étape de seuillage permet également d'être plus précis lors du recalage des profils distance entre eux. Encore une fois, plus le RSB est faible et plus il semble difficile de faire la différence entre les profils distance des différentes classes et plus particulièrement pour les classes 1 et 2 qui, comme nous le confirment ces superpositions, sont des types d'avions de chasse relativement proches.

2.3 Etude statistique des matrices de covariance

Dans cette section, nous allons nous intéresser plus précisément à certaines propriétés statistiques des bases de test et d'apprentissage pour chacune des classes. Les notations utilisées pour la base d'apprentissage sont résumées dans le tableau 2.1. Les notations pour la base de test ne sont pas détaillées car elles sont similaires à celles utilisées pour la base d'apprentissage (indice A pour la base d'apprentissage, indice T pour la base de test).

M	Taille d'un profil distance.
K	Nombre de classes.
N_A^k	Nombre de profils distance de la base d'apprentissage appartenant à la classe k.
$N_A = \sum_{k=1}^{K} N_A^k$	Nombre total de profils distance dans la base d'apprentissage.
$\boldsymbol{X}_A$	Matrice des profils distance de la base d'apprentissage.
$\boldsymbol{x}_{A,i}$	i-ème profil distance de la base d'apprentissage avec $i = [1, \ldots, N_A]$.
$\boldsymbol{X}_A^k$	Matrice des profils distance de la base d'apprentissage appartenant à la classe k.
$\boldsymbol{x}_{A,i}^k$	i-ème profil distance de la base d'apprentissage appartenant à la classe k, avec $k = \{1, \ldots, K\}$ et $i = [1, \ldots, N_A^k]$.
$\boldsymbol{m}_A^k$	Profil distance moyen de la classe k de la base d'apprentissage.
$\boldsymbol{C}_A^k$	Matrice de covariance intra-classe de la classe k de la base d'apprentissage.

TAB. 2.1 : Notations utilisées pour la base d'apprentissage

Le profil distance moyen de la classe k pour la base d'apprentissage est défini de la manière suivante :

$$\boldsymbol{m}_A^k = \frac{1}{N_A^k} \sum_{i=1}^{N_A^k} \boldsymbol{x}_{A,i}^k \tag{2.1}$$

Les éléments de la matrice de covariance $\boldsymbol{C}_A^k$ sont définis de la manière suivante :

$$\boldsymbol{C}_A^k(p,q) = \frac{1}{N_A^k - 1} \sum_{i=1}^{N_A^k} (x_{A,i}^k(p) - m_A^k(p))(x_{A,i}^k(q) - m_A^k(q)) \tag{2.2}$$

Une étude des propriétés statistiques des matrices de covariance intra-classe ($\boldsymbol{C}_A^1$, $\boldsymbol{C}_A^2$, $\boldsymbol{C}_A^3$ et $\boldsymbol{C}_T^1$, $\boldsymbol{C}_T^2$, $\boldsymbol{C}_T^3$) est proposée pour le cas où les profils distance sont récalés et pour les deux configurations suivantes : (i) profils distance seuillés et (ii) profils distance non seuillés.

Cette étude s'articule autour de l'estimation de deux grandeurs :

- les rangs des matrices de covariance ;
- les valeurs propres des matrices de covariance.

Les rangs (nombre maximal de vecteurs lignes (ou colonnes) linéairement indépendants) de ces matrices sont calculés afin d'avoir une idée plus précise de la dimension réelle des données. Les valeurs propres de ces matrices nous renseignent quant à elles sur l'extension des différentes classes. Pour les valeurs propres, on trace le spectre associé et on calcule le produit des deux valeurs propres les plus grandes que l'on note Π_{12} (qui représente le volume de la classe).

2.3.1 Inertie des matrices de covariance

Le spectre des valeurs propres représente le pourcentage d'inertie des valeurs propres de la plus grande à la plus petite. L'inertie $I_{A,i}^k$ de la va-

leur propre λ_i de la matrice de covariance $\boldsymbol{C}_A^k$ est définie de la manière suivante :

$$I_{A,i}^k = \frac{\lambda_i}{\sum_{j=1}^{N_A^k} \lambda_j} \tag{2.3}$$

L'inertie d'une valeur propre peut être vue comme la contribution de cette valeur propre à l'inertie totale de la matrice de covariance. L'inertie est donc une grandeur comprise entre 0 et 1.

Les figures 2.7, 2.8 et 2.9 représentent les sommes cumulées pour les 50 premières valeurs propres pour la base d'apprentissage et la base de test et toujours pour les trois valeurs de RSB différentes. On a positionné également sur ces différentes courbes les pourcentages d'inertie des deux plus grandes valeurs propres.

RSB = 30dB

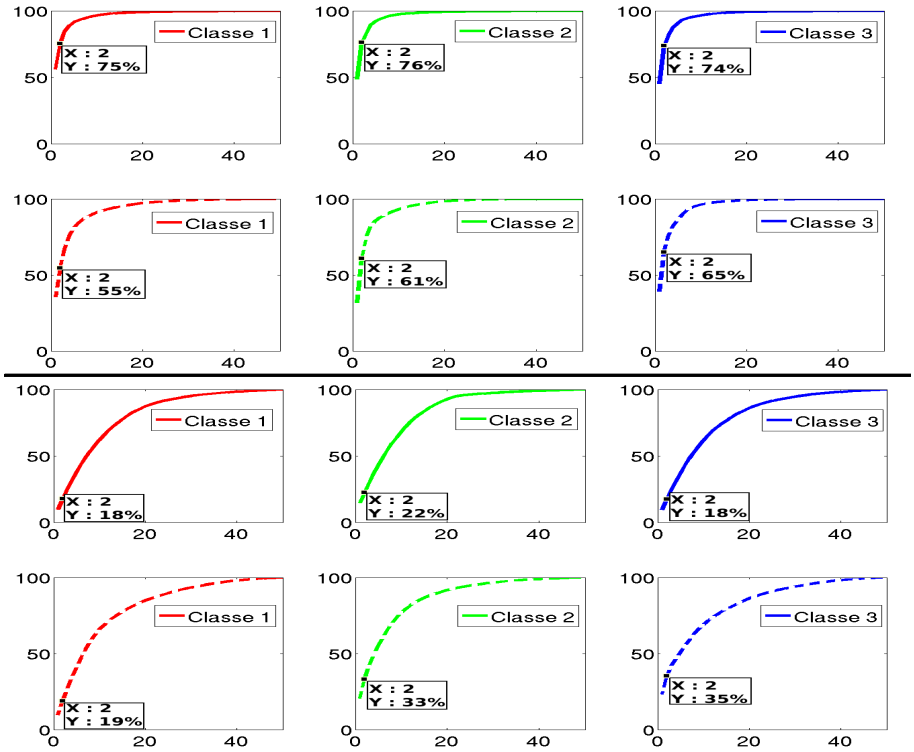


FIG. 2.7 : Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 30dB$

Sur la figure 2.7, on remarque que lorsque les profils distance ne sont pas seuillés, l'information est répartie dans de nombreuses directions de l'espace : l'information contenue par les deux plus grandes valeurs propres représente entre 20% et 30% de l'information totale. Lorsque les profils distance sont seuillés, une grande partie de l'information se concentre sur quelques directions de l'espace. L'inertie des deux plus

grandes valeurs propres se situe entre 55% et 65% pour la base de test et à environ 75% pour la base d'apprentissage. On retrouve des pourcentages d'inertie plus bas pour la base de test, ce qui peut s'expliquer par le fait que l'opération de recalage ne soit pas identique entre la base d'apprentissage et la base de test (base d'apprentissage recalée à partir des profils distance non bruités alors que pour la base de test, l'étape de recalage s'effectue directement sur les profils distance bruités).

RSB = 20dB

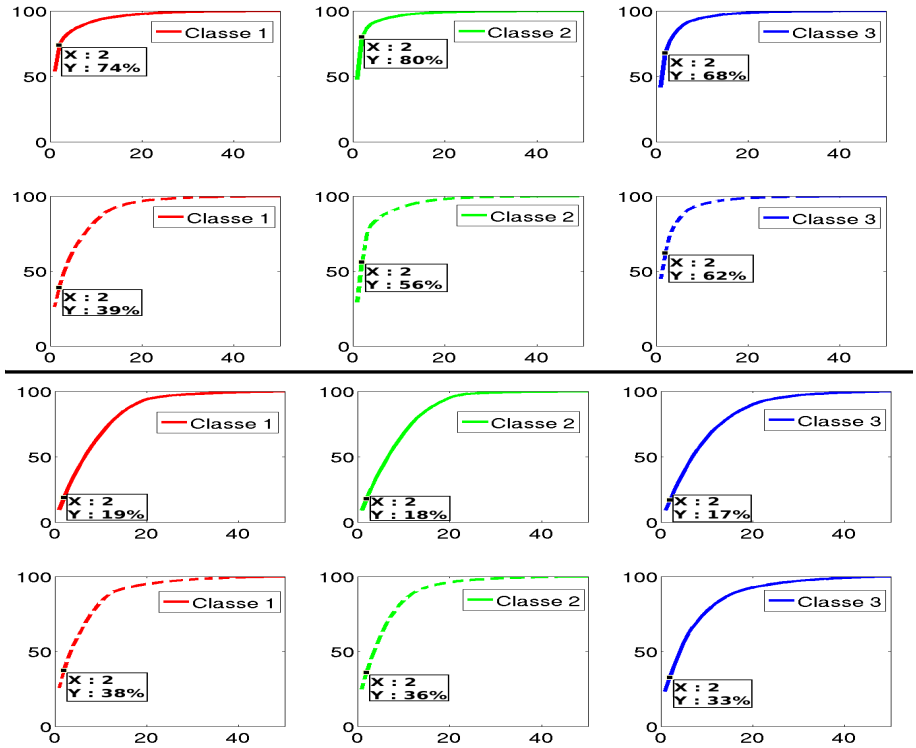


FIG. 2.8 : Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 20dB$

Sur la figure 2.8, on retrouve la tendance que l'on a observée avec un $RSB = 30dB$. On se rend compte tout de même que l'information contenue par les deux plus grandes valeurs propres pour les profils distance de la base de test seuillés diminue, notamment pour la classe 1.

RSB = 15dB

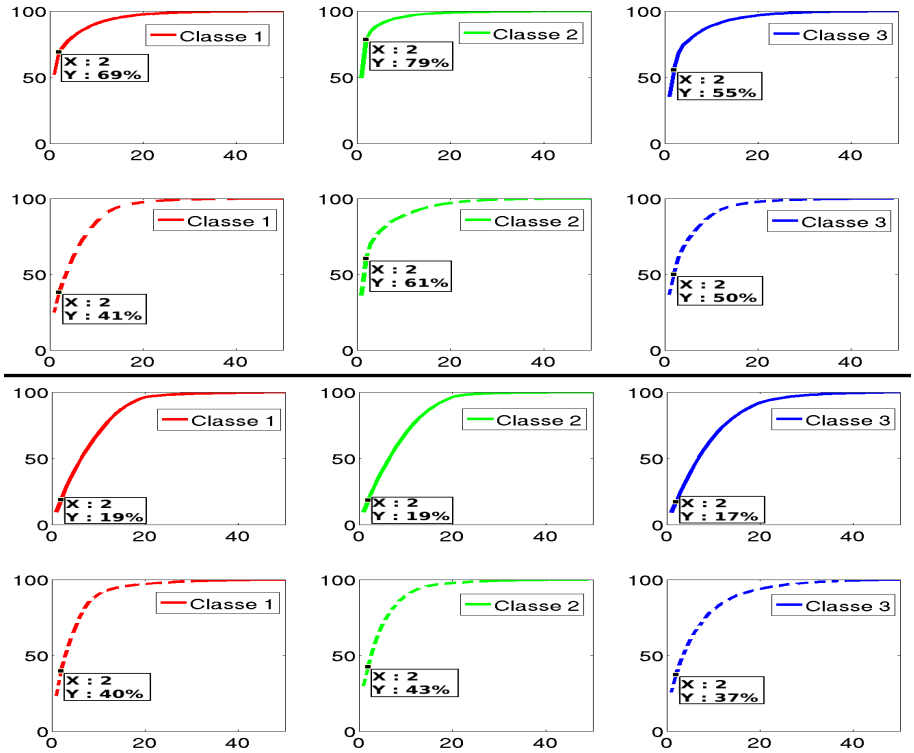


FIG. 2.9 : Somme cumulée des 50 premières valeurs propres de la base d'apprentissage (trait continu) et de la base de test (trait pointillé) pour des profils distance seuillés (en haut) et non seuillés (en bas) pour un $RSB = 15dB$

D'une manière générale, le pourcentage d'inertie des deux plus grandes valeurs propres à tendance à diminuer plus le RSB diminue. L'idée de caractériser le volume des classes par les deux plus grandes valeurs propres apparaît donc de plus plus en plus approximative au fur et à mesure que le RSB diminue. Cependant avec un pourcentage d'inertie minimum situé aux alentours de 40%, cela reste tout de même une bonne manière d'évaluer le volume des classes et d'avoir une approximation de leur répartition dans un espace 2D.

2.3.2 Volume des classes

Les tableaux 2.2, 2.4 et 2.6 résument les valeurs de Π_{12} et des rangs des matrices de covariance pour les profils distance de la base d'apprentissage et de test seuillés et les tableaux 2.3, 2.5 et 2.7 pour les profils distance de la base d'apprentissage et de test non-seuillés et cela pour les trois niveaux de RSB.

RSB = 30dB

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_A^1	3.7e4	189	1.7e6	341
C_A^2	1.2e5	202	3.3e6	341
C_A^3	1.3e4	109	1.5e6	341

TAB. 2.2 : Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 30dB$

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_T^1	1.8e4	49	1.7e6	49
C_T^2	2.3e5	49	8.8e6	49
C_T^3	1.1e4	49	1.1e7	49

TAB. 2.3 : Caractérisation statistique des profils distance de la base de test pour un $RSB = 30dB$

Au vu des valeurs de Π_{12}, il semble que la classe 2 soit la plus étendue. Les rangs des matrices de covariance montrent que la dimension réelle des données peut être réduite lorsque les profils distance sont seuillés. On peut voir également que lorsque les profils distance ne sont pas seuillés, le volume des classes est beaucoup plus grand, ce qui peut rendre la classification plus complexe. Ces observations restent valables pour la base de test. En revanche, les matrices de covariance restent de rang plein même avec les profils distance seuillés. Cela peut s'expliquer par le fait que le nombre de profils distance de test est beaucoup moins important que celui de la base d'apprentissage et donc les possibilités de combinaisons linéaires entre les lignes ou les colonnes sont beaucoup moins importantes. Une deuxième explication peut venir une nouvelle fois du fait que le recalage des profils distance de test est effectué sur des données bruitées au contraire de la base d'apprentissage.

RSB = 20dB

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_A^1	8.7e3	312	1.9e6	341
C_A^2	4.5e4	287	1.8e6	341
C_A^3	4.2e3	173	1.5e6	341

TAB. 2.4 : Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 20dB$

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_T^1	3.7e3	49	6.1e6	49
C_T^2	4.9e4	49	5.4e6	49
C_T^3	4.4e3	49	5.3e6	49

TAB. 2.5 : Caractérisation statistique des profils distance de la base de test pour un $RSB = 20dB$

Avec un $RSB = 20dB$, le volume des classes a tendance à diminuer légèrement du fait de l'augmentation du niveau de bruit et donc de l'augmentation du niveau de seuillage. Les rangs des matrices de covariance ont également augmenté.

RSB = 15dB

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_A^1	2.7e3	303	1.9e6	341
C_A^2	1.6e4	297	1.9e6	341
C_A^3	1.1e3	271	1.6e6	341

TAB. 2.6 : Caractérisation statistique des profils distance de la base d'apprentissage pour un $RSB = 15dB$

	Seuillés		Non-seuillés	
	Π_{12}	Rg	Π_{12}	Rg
C_T^1	1.3e3	49	5.2e6	49
C_T^2	1.5e4	49	4.5e6	49
C_T^3	2.3e3	49	4.8e6	49

TAB. 2.7 : Caractérisation statistique des profils distance de la base de test pour un $RSB = 15dB$

Avec un $RSB = 15dB$, les observations restent les mêmes que celles faites jusqu'à présent. L'étude de ces grandeurs statistiques nous a permis d'obtenir une estimation du volume des différentes classes. Il apparaît que la classe 2 est la classe la plus étendue. Le seuillage des profils distance a tendance à réduire le volume des différentes classes, ce qui peut rendre la classification plus aisée. Dans la suite de ce chapitre, nous allons chercher à obtenir une visualisation de la disposition des classes les unes par rapport aux autres, toujours en utilisant l'information contenue par les deux plus grandes valeurs propres.

2.3.3 Représentation graphique des données

Une solution simple pour représenter les données dans un espace 2D consiste à visualiser seulement les deux directions contenant le plus d'information. On peut calculer la distance entre les profils distance moyens des classes 1, 2 et 3 (cf. tableaux 2.8, 2.10, 2.12 et 2.9, 2.11, 2.13). La dernière ligne de ces tableaux est une représentation graphique des grandeurs calculées jusqu'à présent dans cette section. Il s'agit en fait de représenter les centres des trois classes sur un triangle dont les cotés sont de tailles égales aux distances calculées dans les tableaux 2.8, 2.10, 2.12 et 2.9, 2.11, 2.13. Comme on l'a déjà mentionné avant, on considère que l'information de volume des différentes classes peut être approximée à partir des deux plus grandes valeurs propres. La classe est donc caractérisée par une ellipse avec comme valeur de grand axe la première valeur propre et comme valeur de petit axe la seconde valeur propre. La direction du grand axe de l'ellipse est donnée par la direction du vecteur propre correspondant à la plus grande valeur propre. On fait l'hypothèse que les vecteurs propres des deux plus grandes valeurs propres sont orthogonaux. Il s'agit donc au final d'une ACP avec 2 composantes. On peut définir la distribution associée de la manière suivante :

$$f_i(\boldsymbol{x}) = 2\pi|\boldsymbol{\Sigma}_i|e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i)} \tag{2.4}$$

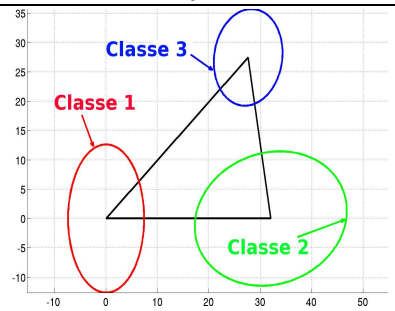
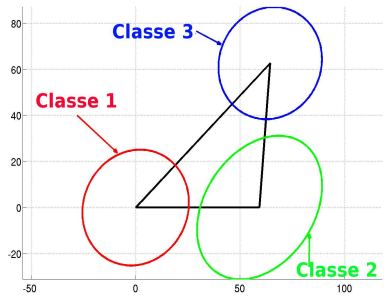
avec :

$$\boldsymbol{\Sigma}_i^{-1} = \begin{pmatrix} \lambda_i^1 & 0 \\ 0 & \lambda_i^2 \end{pmatrix} \tag{2.5}$$

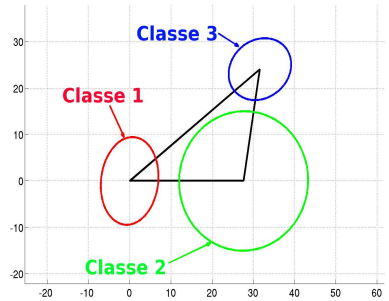
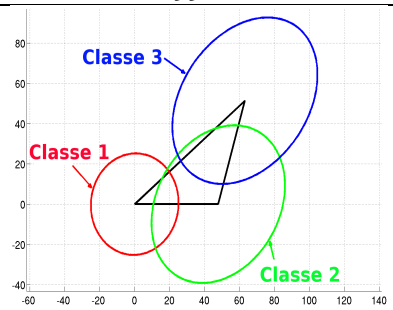
et m_i la moyenne de la classe i et λ_i^1 et λ_i^2 les deux plus grandes valeurs propres de la classe i.

Les ellipses à 95% associées à chaque classe sont donc tracées sur la dernière ligne des tableaux 2.8 et 2.9 pour un $RSB = 30dB$, des tableaux 2.10 et 2.11 pour un $RSB = 20dB$ et des tableaux 2.12 et 2.13 pour un $RSB = 15dB$.

RSB=30dB

	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	32.1	59.3
$1 \leftrightarrow 3$	38.9	89.9
$2 \leftrightarrow 3$	27.8	62.9
		

TAB. 2.8 : Distance euclidienne entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 30dB$

	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	27.6	47.8
$1 \leftrightarrow 3$	39.7	81.3
$2 \leftrightarrow 3$	24.4	53.6
		

TAB. 2.9 : Distance euclidienne entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 30dB$

On observe sur le tableau 2.8 et le tableau 2.9 que les classes semblent relativement bien séparées même dans le cas où les profils distance de la base d'apprentissage ne sont pas seuillés (ceci n'est pas le cas des profils distance de test non seuillés). On peut noter qu'après seuillage, les classes semblent tout de même plus éloignées les unes des autres et donc plus facilement différentiables. Afin de se faire une idée plus précise des performances atteignables, il peut être intéressant de superposer les distributions des différentes classes pour la base d'apprentissage et de test. Plus précisemment, nous superposons sur la figure 2.10, les ellipses à 95% des distributions des classes 1, 2 et 3 pour les bases d'apprentissage et de test dans le cas seuillé et non-seuillé. Pour superposer les deux courbes, on

place les deux triangles de manière à ce que leurs centres de gravité soient à l'origine du repère. Ensuite pour pouvoir exploiter cette figure correctement, on cherche la position d'équilibre de ces deux triangles dans le repère (X, Y). Il s'agit d'un problème de mécanique qui consiste à trouver la position du triangle pour laquelle la somme des moments en son centre de gravité est nulle, en considèrant trois forces de norme équivalente s'exerçant aux trois sommets et dirigées en $-Y$.

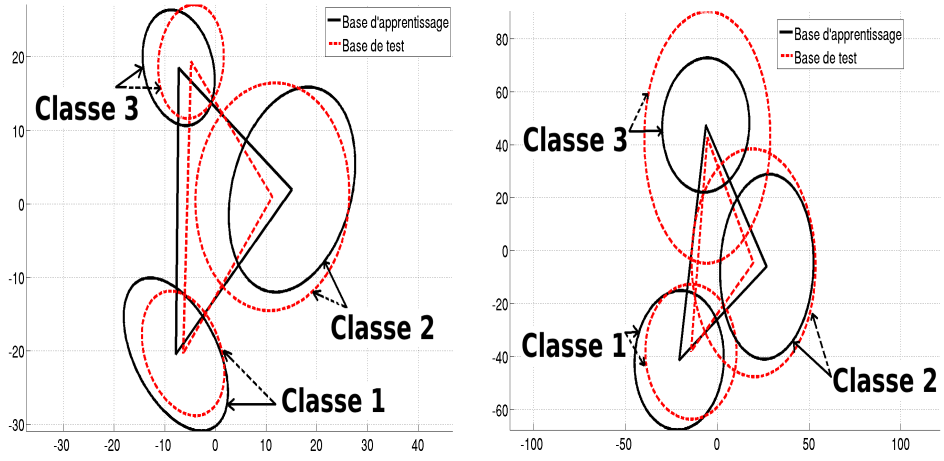


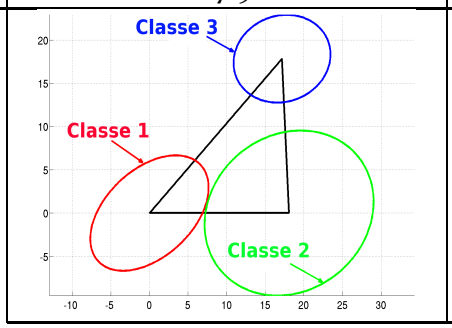
FIG. 2.10 : Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 30dB$

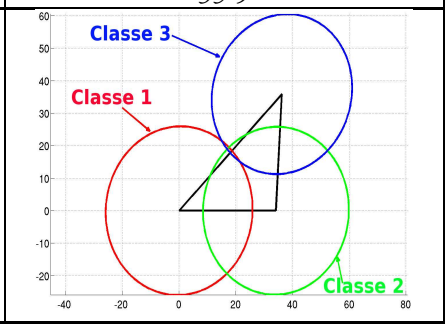
On peut voir sur la figure 2.10 que :

- Pour les profils distance seuillés, les ellipses de la base d'apprentissage et de la base de test pour chaque classe sont proches. Entre les différentes classes, les ellipses ne se chevauchent pas.
- Pour les profils distance non seuillés, les ellipses entre classe se chevauchent.

RSB=20dB

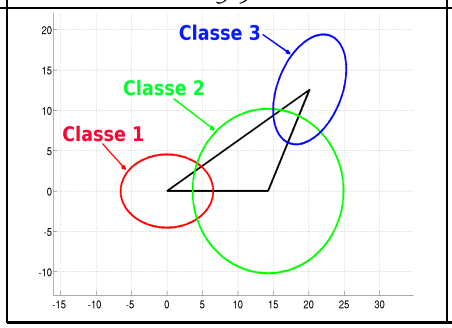
	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	18.1	34.1
$1 \leftrightarrow 3$	24.7	51.0
$2 \leftrightarrow 3$	17.9	35.9

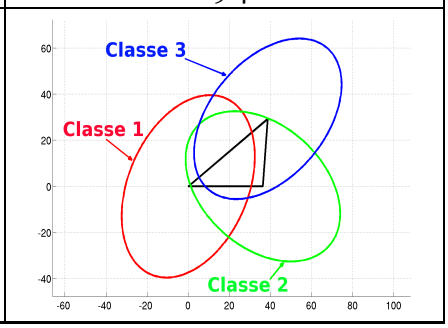




TAB. 2.10 : Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 20dB$

	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	14.2	36.3
$1 \leftrightarrow 3$	23.7	48.5
$2 \leftrightarrow 3$	13.9	29.4





TAB. 2.11 : Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 20dB$

Avec un RSB plus faible (cf. tableaux 2.10 et 2.11), on s'aperçoit que les ellipses des classes 1 et 2 ont tendance à se rapprocher pour la base d'apprentissage et on constate des chevauchements pour la base de test entre les classes 1 et 2 et les classes 2 et 3. On retrouve encore une fois cette différence entre profils distance de test et d'apprentissage que l'on peut toujours expliquer par l'étape de recalage des profils distance différents entre les deux bases.

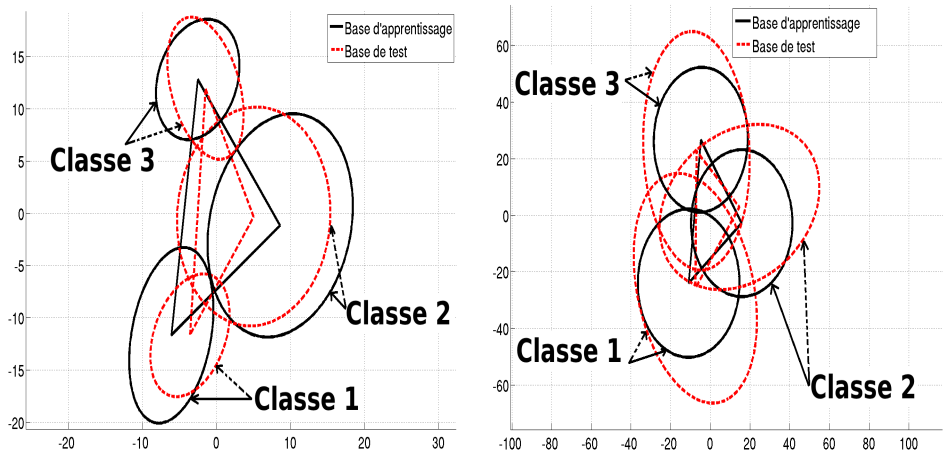
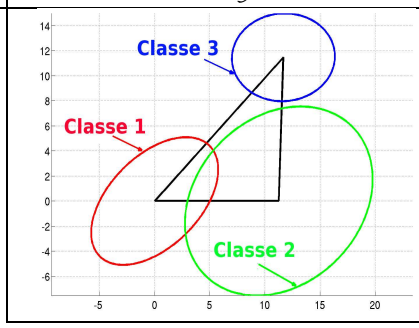
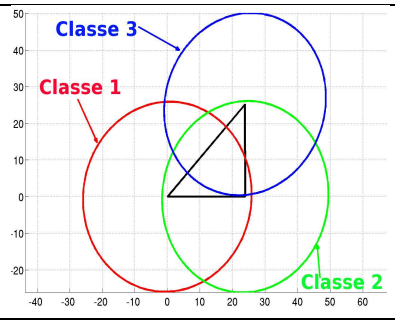


FIG. 2.11 : Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 20dB$

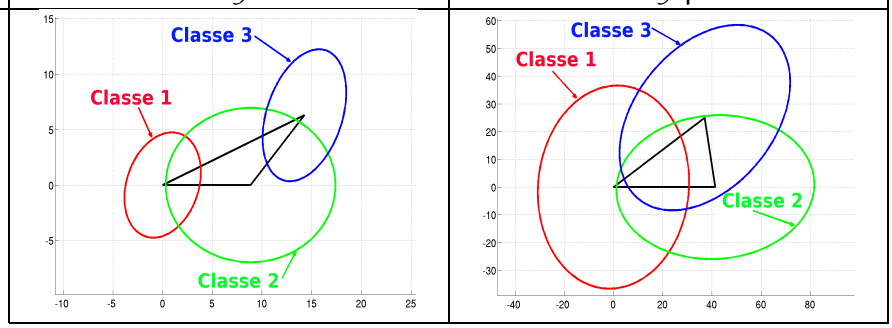
On peut voir sur la figure 2.11 que la superposition devient moins nette. L'ellipse de la classe 1 de la base de test chevauche légèrement l'ellipse de la classe 2 de la base d'apprentissage, ce qui peut présager d'une difficulté à discriminer ces deux classes. Cette prévision est renforcée par le fait que pour la classe 2, l'ellipse de test chevauche l'ellipse d'apprentissage de la classe 1.

RSB=15dB

	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	11.2	23.9
$1 \leftrightarrow 3$	16.3	34.7
$2 \leftrightarrow 3$	11.5	25.8
		

TAB. 2.12 : Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base d'apprentissage pour un $RSB = 15dB$

	Seuillés	Non-seuillés
	Distance L2 entre les moyennes	Distance L2 entre les moyennes
$1 \leftrightarrow 2$	8.8	41.4
$1 \leftrightarrow 3$	15.6	44.8
$2 \leftrightarrow 3$	8.3	25.4



TAB. 2.13 : Distance euclidienne normalisée entre les profils distance moyens de la classe 1,2 et 3 de la base de test pour un $RSB = 15dB$

On observe sur les tableaux 2.12 et 2.13 qu'à présent les ellipses d'apprentissage des classes 1 et 2 se chevauchent et il existe toujours un chevauchement entre les classes 1 et 2 et classes 2 et 3 pour la base de test.

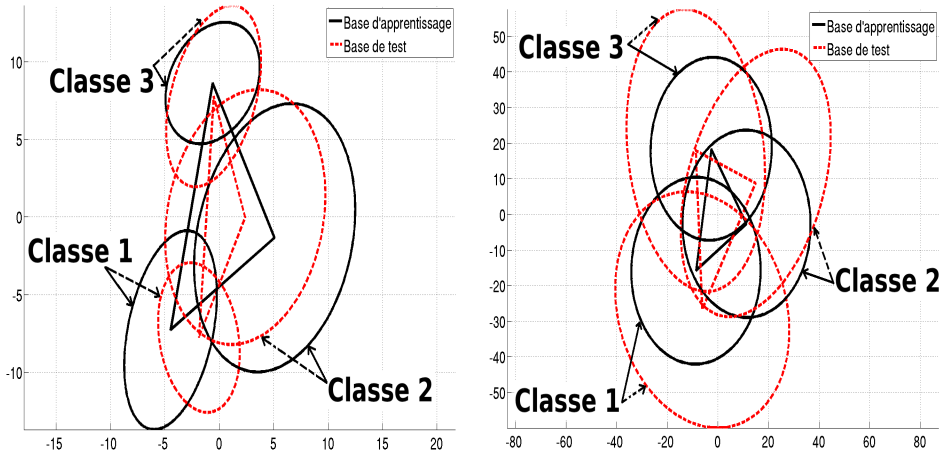


FIG. 2.12 : Superposition ellipse à 95% pour la base de test et la base d'apprentissage avec des profils distance seuillés (à gauche) et non-seuillés (à droite) pour un $RSB = 15dB$

A partir de la figure 2.12, on peut dire que plus le RSB diminue, plus il y a de chevauchements entre les classes. Or, plus le chevauchement entre deux classes est important, plus le risque qu'un profil distance soit affecté à une autre classe est grand. Il y a donc plus de risque d'avoir un taux d'erreur important. Sous ces conditions de RSB, il semble que le risque d'erreur de classification, particulièrement entre les classes 1 et 2 et dans une moindre mesure entre les classes 2 et 3, soit élevé.

2.4 Conclusion du chapitre

Cette étude de la base de données permet de manière assez simple d'avoir une vision plus précise des données utilisées et des difficultés que nous pourrons rencontrer, notamment à cause de certaines propriétés de celles-ci, lors du développement de nos algorithmes. Bien qu'il ne s'agisse que d'une étude approximative, elle permet de se faire une idée de la manière dont se répartissent les données et notamment d'en obtenir une visualisation dans un espace 2D. Cette étude a permis en outre de mettre en évidence l'importance du seuillage des profils distance pour éliminer les échantillons de bruit et faciliter la classification des données. Dans la suite du mémoire, nous allons proposer et étudier plusieurs types d'algorithmes permettant de faire de la reconnaissance de cibles non coopératives. Nous utiliserons ces bases de données pour évaluer les performances de nos algorithmes, il sera donc toujours intéressant d'observer le comportement de nos algorithmes en se reportant à cette étude.

Algorithme des KPPV

3

Sommaire

3.1 Présentation générale 62
3.2 Première étape : calcul des métriques 62
3.3 Deuxième et troisième étapes : tri des métriques calculées à l'étape 1 et décision 63
3.4 Paramètres de réglage 64
3.5 Performances de l'algorithme des KPPV 64
3.5.1 $RSB = 30dB$ 65
3.5.2 $RSB = 20dB$ 65
3.5.3 $RSB = 15dB$ 66
3.5.4 Matrices d'acceptation (MA) ou matrice de confusion d'acceptation 66
3.6 Conclusion du chapitre 67

L'idée de base des problématiques NCTR consiste à comparer la signature radar de la cible à identifier avec des signatures contenues dans une base d'apprentissage. On peut donc assimiler le problème NCTR à un problème de classification supervisée. Parmi ces méthodes, on trouve la méthode des K Plus Proches Voisins (KPPV). Cette méthode va classer la cible dans la classe majoritaire parmi les K plus proches voisins. Dans ce chapitre, nous présentons donc une étude de cet algorithme vis-à-vis des contraintes imposées par les applications NCTR. Le choix de cet algorithme par rapport à d'autres techniques de classification supervisée est justifié par notre intention de ne pas compresser l'information et d'évaluer les performances d'algorithmes de type "force-brute", alternative rendue possible dans une optique de traitement temps-réel depuis l'arrivée en 2006 des processeurs "many-cores" de type GPU. Nous nous intéressons dans ce chapitre aux performances de l'algorithme des KPPV sur la base de données synthétiques et nous étudions l'influence que peuvent avoir les différents paramètres de réglages sur les performances de classification. La section 3.1 est une présentation générale de l'algorithme des KPPV et des différentes étapes qui le composent. Nous revenons ensuite en détail sur ces différentes étapes, notamment l'étape de calcul des métriques dans la section 3.2 puis le tri des métriques calculées et la prise de décision dans la section 3.3. La section 3.4 résume les différents paramètres

de réglages de l'algorithme des KPPV. Au final, nous présentons dans la section 3.5, les résultats de l'algorithme avec différentes configurations sur les paramètres (nombre de plus proches voisins, type de métriques, type de prétraitements) puis nous concluons sur les avantages et les défauts de l'algorithme des KPPV pour une utilisation dans le contexte NCTR.

3.1 Présentation générale

L'algorithme des KPPV est fondé sur une méthode d'apprentissage supervisé (Duda et al. 2001, Kotsiantis 2007, Kantardzic 2011). Soit $\boldsymbol{x}_T$ et $\boldsymbol{X}_A$, respectivement le profil distance sous test et la base d'apprentissage. $\boldsymbol{x}_T$ est un vecteur de taille N échantillons. $\boldsymbol{X}_A$ est une matrice $N_A \times N$ avec N_A, le nombre de signatures dans la base d'apprentissage. Les différentes étapes de l'algorithme des KPPV sont représentées sur la la figure 3.1.

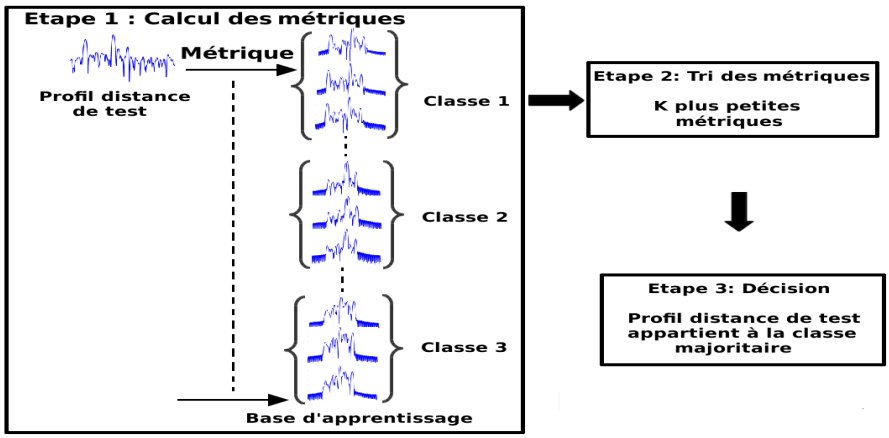


Fig. 3.1 : Principe de l'algorithme des KPPV

3.2 Première étape : calcul des métriques

La première étape de l'algorithme des KPPV consiste à calculer les métriques entre $\boldsymbol{x}_T$ et les N_A profils distance de la base d'apprentissage $\boldsymbol{X}_A$. Nous avons étudié plusieurs types de métriques :

- les distances : ce sont des métriques qui respectent les conditions de symétrie, de séparation et d'inégalité triangulaire.
- les semimétriques, semi-distances : ce sont en général des métriques issues des coefficients de corrélation. Ce ne sont pas des distances car elles ne respectent pas l'inégalité triangulaire.

Pour cette étude, deux distances ont été étudiées : la distance euclidienne (L2) et la distance de Manhattan (L1). Ces deux distances sont définies de la manière suivante :

- Distance euclidienne (L2) : c'est la distance la plus souvent utilisée. Son expression est la suivante :

$$d_{euclidian} = \sqrt{\sum_i (x_i - y_i)^2} \tag{3.1}$$

– Distance de Manhattan (L1) : cette distance permet d'être plus robuste aux valeurs abérrantes. Si la distance euclidienne correspond à la longueur du chemin le plus court entre deux points, la distance de Manhattan est quant à elle la somme des distances le long de chaque dimension. Elle est définie de la manière suivante :

$$d_{manhattan} = \sum_i |x_i - y_i| \tag{3.2}$$

Nous avons également étudié deux semimétriques basées sur les coefficients de corrélation de Pearson et de Spearman :

– Distance de Pearson : elle est définie ainsi $d_{pearson} = 1 - r_p$ avec r_p le coefficients de corrélation de Pearson :

$$r_p = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i - \bar{x}}{\sigma_x} \right) \left(\frac{y_i - \bar{y}}{\sigma_y} \right) \tag{3.3}$$

avec $\bar{x}$, $\bar{y}$ respectivement la moyenne sur les x_i et y_i et σ_x, σ_y l'écart type sur les x_i et y_i.
Le coefficient de corrélation de Pearson est un coefficient de corrélation linéaire. Cherchant une liaison dans une distribution à deux variables, nous établissons une régression linéaire simple, résumant graphiquement un nuage de points par une droite, dite de régression. La qualité de cette régression est mesurée par le coefficient de corrélation de Pearson. C'est la covariance entre la variable explicative x et la variable à expliquer y, rapportée au produit de leurs écarts-types.

– Distance de Spearman : elle est définie ainsi $d_{spearman} = 1 - r_s$ avec r_s le coefficients de corrélation de Spearman. Le coefficient de corrélation de Spearman est défini comme étant le coefficient de Pearson non pas sur les variables directement mais sur leurs mesures de rang (r_{x_i},r_{y_i}).

$$r_s = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{r_{x_i} - \bar{r_x}}{\sigma_{r_x}} \right) \left(\frac{r_{y_i} - \bar{r_y}}{\sigma_{r_y}} \right) \tag{3.4}$$

Cette corrélation est utilisée lorsque deux variables semblent corrélées sans que leur relation soit de type affine. Cela correspond au cas où les distributions des variables ne sont pas Gaussiennes mais plutôt asymétriques.

3.3 Deuxième et troisième étapes : tri des métriques calculées à l'étape 1 et décision

La seconde étape consiste à trier les valeurs des métriques obtenues à l'étape 1. On note $\boldsymbol{d}_K$ le vecteur contenant les K plus petites métriques.

Dans l'étape de décision, le nombre d'occurences de chacune des trois classes que contient le vecteur $\boldsymbol{d}$ est déterminé. Le profil distance sous test est classé dans la classe majoritaire parmi les K plus proches voisins. En

cas d'égalité, le profil distance sera affecté à la classe possédant le voisin le plus proche du profil distance sous test.

3.4 Paramètres de réglage

A partir de cette description, il apparaît que trois paramètres peuvent influencer le résultat de l'algorithme des KPPV :

- le type de métriques utilisées
- le nombre de plus proches voisins K
- la nature des signatures utilisées et les prétraitements : pour calculer les métriques entre les profils distance, nous devons nous assurer que les échantillons de bruit ne sont pas pris en compte. Les signaux doivent donc être débruités en amont du calcul de la métrique par une étape de seuillage. Comme on l'a mentionné dans la section 2.1.1, les profils distance sont des données bruitées. Pour déterminer le seuil, le signal utile se distinguant assez facilement du bruit sur un profil distance grâce au gain de compression d'impulsion, on calcule la puissance du bruit sur le profil distance de test et d'apprentissage, puis on fixe le seuil à partir de la valeur maximum de la puissance de bruit entre le profil distance de test et d'apprentissage. Le seuil peut donc être différent pour chaque profil distance de test et d'apprentissage. A noter, que l'on peut également éliminer les échantillons du bruit en appliquant une fenêtre temporelle sur le profil distance plutôt que d'effectuer un seuillage en amplitude. Cette solution donne des résultats comparables à ceux obtenus en seuillant les profils distance en amplitude.

3.5 Performances de l'algorithme des KPPV

Pour évaluer les performances, nous traçons le taux d'erreur en fonction de K avec les profils distance seuillés et non seuillés, pour les quatre métriques définies dans la section 3.2 et pour trois valeurs de RSB différentes ($RSB = 30dB$, $RSB = 20dB$, $RSB = 15dB$). Le taux de succès étant le complément à 1 du taux d'erreur, il se déduit donc facilement du taux d'erreur et il n'est pas nécessaire de le représenter.

3.5.1 $RSB = 30dB$

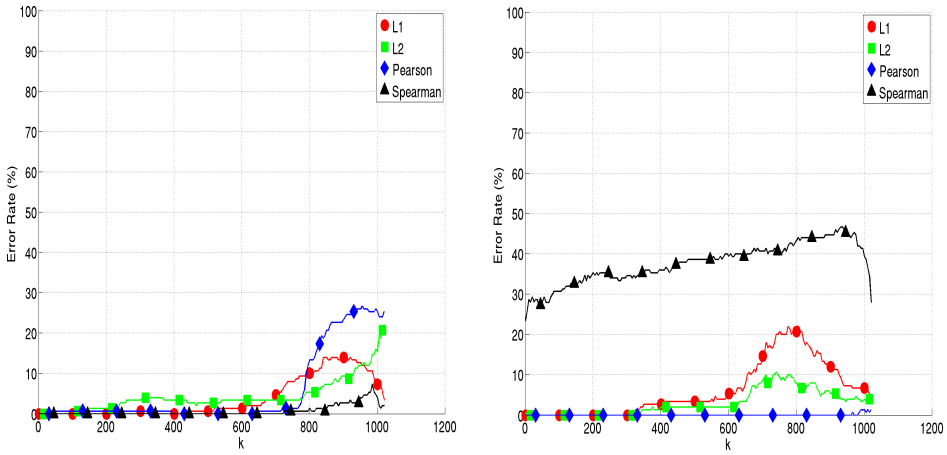


FIG. 3.2 : Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 30dB$)

La figure 3.2 montre que les performances de l'algorithme sans seuillage des profils distance sont très mauvaises avec la distance de Spearman. Cependant lorsque les profils distance sont seuillés, les résultats sont à peu près équivalents avec les différentes métriques. On s'aperçoit également que le nombre de K plus proches voisins n'a pas une grande influence sur les performances tant qu'il reste suffisamment faible.

3.5.2 $RSB = 20dB$

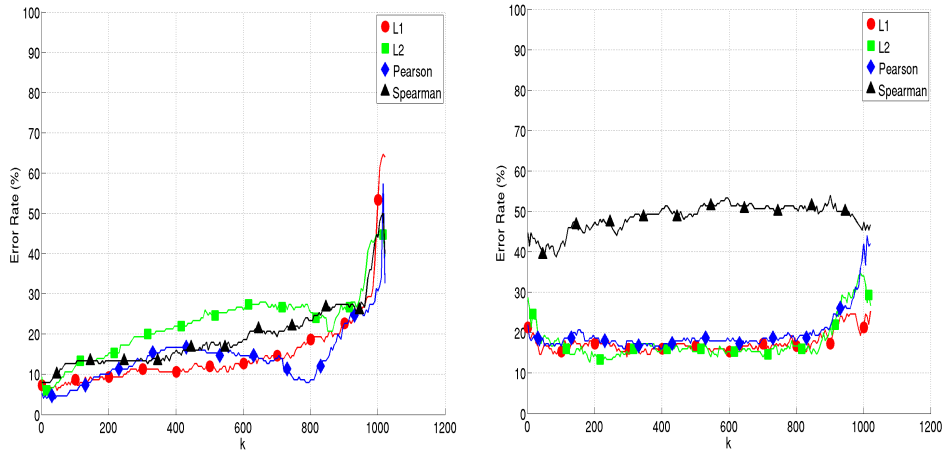


FIG. 3.3 : Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 20dB$)

La figure 3.2 nous permet encore une fois de montrer que la distance de Spearman n'est pas bien adaptée aux profils distance non seuillés. On

s'aperçoit également que le taux d'erreur augmente très nettement dans le cas sans seuillage et cela quel que soit le nombre de plus proches voisins. Par contre, en seuillant les profils distance et pour de faibles valeurs de K, le taux d'erreur reste acceptable ($\approx$ 5%) quelle que soit la métrique utilisée.

3.5.3 $RSB = 15dB$

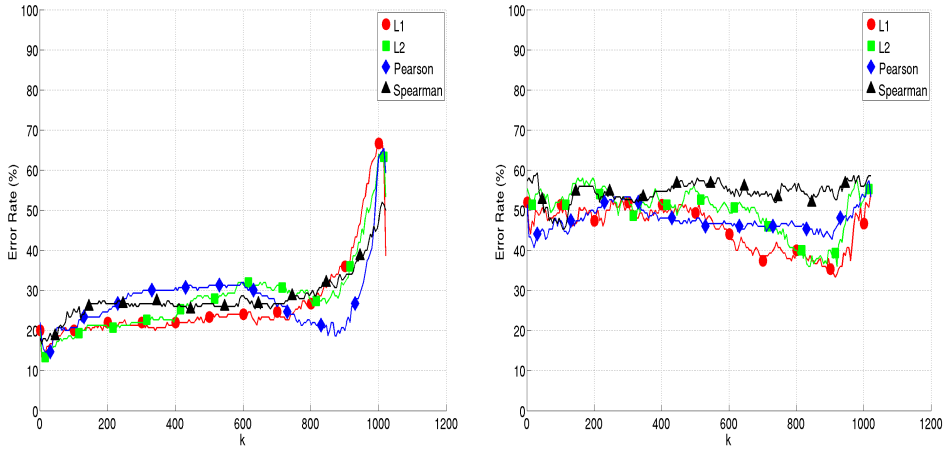


FIG. 3.4 : Taux d'erreur en fonction de K pour l'algorithme des KPPV avec seuillage (gauche) et sans seuillage (droite) des profils distance ($RSB = 15dB$)

Avec un $RSB = 15dB$ (cf. figure 3.4), le taux d'erreur commence à se dégrader considérablement même en seuillant les profils distance ($\approx$ 15%). Le choix de la métrique ou le nombre de plus proches voisins K ne permet pas de réduire le taux d'erreur. Celui-ci devient trop élevé pour le contexte NCTR où une erreur peut entraîner la perte de vies humaines.

3.5.4 Matrices d'acceptation (MA) ou matrice de confusion d'acceptation

En plus de calculer les taux d'erreur, on peut également étudier la matrice d'acceptation (aussi appelée matrice de confusion d'acceptation), que l'on notera MA dans la suite de notre rapport. Les lignes de la matrice correspondent aux profils distance de test des trois différentes classes. Les colonnes correspondent à la classe dans laquelle ils ont été classés. Les valeurs affichées sont des pourcentages. Pour mieux comprendre, prenons par exemple la première ligne du tableau 3.1. Il s'agit donc des profils de test de la classe 1. La première colonne nous dit que 100% de ces profils ont été classés dans la classe 1, la deuxième colonne nous dit que 0% ont été classés dans la classe 2 et la troisième colonne que 0% ont été classés dans la classe 3. Le tableau 3.1 regroupe les matrices d'acceptation pour les trois niveaux de RSB. Au vu des courbes obtenues dans la section précédente, nous présentons les matrices de confusion avec comme paramètres un nombre de plus proches voisins $K = 5$, avec des profils

distance seuillés, en utilisant la distance euclidienne (L2) comme métrique entre les profils distance, ce qui correspond aux paramètres permettant d'obtenir les meilleures performances.

		MA		
		1	2	3
$RSB = 30dB$	1	100	0	0
	2	0	100	0
	3	0	0	100
$RSB = 20dB$	C1	96	4	0
	C2	14	86	0
	C3	2	0	100
$RSB = 15dB$	C1	94	4	2
	C2	28	70	2
	C3	6	0	94

Tab. 3.1 : Matrice d'acceptation pour les trois niveaux de RSB - Algorithme des KPPV

Lorsque le RSB est suffisamment élevé, les trois classes sont plutôt bien discriminées entre elles. En revanche, il apparaît assez clairement, dès lors que le RSB diminue, que les classes 1 et 2 sont de plus en plus difficilement distingables alors que la classe 3 reste toujours plutôt bien discriminée. Plus précisément, avec cet algorithme, les profils distance de la classe 2 ont tendance à être classés dans la classe 1. Cette tendance n'est pas surprenante au vu des résultats obtenus dans le chapitre 2.

	TE	TS
RSB=30dB	0	100
RSB=20dB	6.6	93.4
RSB=15dB	14	86

Tab. 3.2 : Taux d'erreur et de succès - Algorithme des KPPV

Le tableau 3.2 est un résumé des performances obtenues en terme de taux d'erreur (TE) et de taux de succès (TS) sous les trois conditions de RSB de notre base de données. Sous un $RSB = 15dB$, ce qui se rapproche le plus des conditions réelles, l'algorithme des KPPV ne permet pas de garantir un taux d'erreur suffisamment faible. Ce point là est cruciale et même si sous des conditions de RSB plus favorables, l'algorithme se comporte plutôt bien, cela n'est pas suffisant pour considérer l'algorithme des KPPV comme une solution viable pour faire de la reconnaissance de cibles non coopératives.

3.6 Conclusion du chapitre

Ce chapitre détaille l'application d'un algorithme de type KPPV pour la reconnaissance de cibles. Un taux d'erreur maîtrisé et un taux de suc-

cès maximisé constitue la première contrainte forte de ce type d'applications. L'algorithme des KPPV a de bonnes performances lorsque le RSB est suffisamment élevé. Par contre, le taux d'erreur augmente lorsque le RSB diminue. Le problème avec l'algorithme des KPPV est qu'il n'intègre aucun mécanisme pour contrôler le taux d'erreur. En effet, de nombreux systèmes de classification pourraient se contenter d'un taux d'erreur de 14% mais dans le contexte NCTR, ce taux est déjà beaucoup trop élevé. Cette étude sur l'algorithme des KPPV nous a quand même permis de tirer un certain nombre d'enseignements que nous pourrons transposer aux autres algorithmes étudiés durant la thèse. Ces enseignements sont :

- Le seuillage des profils distance est indispensable pour pouvoir obtenir de bonnes performances de classification.
- Le type de métriques utilisées n'influe que très peu sur les résultats. L'utilisation de la distance euclidienne (L2) est une bonne solution pour mesurer la similarité entre profils distance.

Pour améliorer les performances, notamment dans la maîtrise du taux d'erreur, on peut imaginer l'utilisation d'autres types de classifieurs. Les classifieurs de type probabiliste ou basés sur la logique floue sont des solutions possibles. Ils sont décrits dans les deux chapitres suivants.

4 Algorithmes de reconnaissance de cibles basés sur les méthodes probabilistes

Sommaire

4.1 Modèle de mélange pour les applications NCTR 70
4.1.1 Gaussian Mixture Models (GMM) 70
4.1.2 Modèle de mélange de Student 71
4.2 Estimation des paramètres et calcul des probabilités *a posteriori* 72
4.2.1 Estimation des paramètres 72
4.2.2 Calcul des probabilités *a posteriori* 73
4.3 Règles de décision 74
4.3.1 Règle de décision du MAP 74
4.3.2 Règle de décision pour le contrôle du taux d'erreur 75
4.4 Paramètres de réglages 75
4.5 Performances de l'algorithme basé sur les méthodes probabilistes 76
4.6 Représentation graphique des données via les méthodes probabilistes 78
4.6.1 Stochastic Neighbor Embedding (SNE) 78
4.6.2 Carte de probabilité dans l'espace de dimension réduite . 80
4.7 Conclusion du chapitre 81

Le but de ce chapitre est d'évaluer les méthodes de type probabiliste pour une application au contexte NCTR et de comparer leurs performances par rapport à l'algorithme des KPPV. L'avantage des méthodes probabilistes est qu'elles permettent d'obtenir une mesure quantitative (la probabilité *a posteriori*) quant à l'appartenance d'un profil distance à une classe, ce qui n'est pas le cas avec l'algorithme de type KPPV. Cela peut permettre notamment de construire des règles de décision à plusieurs niveaux (et non plus binaire comme c'était le cas avec l'algorithme des KPPV), qui

peuvent permettre d'avoir une meilleure maîtrise du taux d'erreur. En revanche, l'estimation des paramètres des distributions, qui plus est dans des espaces de grandes dimensions, oblige le plus souvent à faire des approximations et à réduire la dimension du problème contrairement à l'algorithme des KPPV, ce qui risque de dégrader les performances de classification. En tenant compte de ces remarques, nous proposons dans la section 4.1.2 un algorithme basé sur un modèle de mélange de distributions de Student qui permet une prise en compte des incertitudes sur les données. Dans la section 4.1, nous introduisons d'une manière générale les modèles de mélanges quelconques puis nous nous attardons plus précisement sur les modèles de mélanges de Gaussiennnes (cf. section 4.1.1). Dans la section 4.2, nous abordons le problème de l'estimation des paramètres du modèle et du calcul des probabilités *a posteriori*. Nous présentons ensuite les deux règles de décision utilisées, à savoir la règle classique du maximum *a posteriori* et une règle de décision adaptée au contexte NCTR qui a été définie dans l'optique de pouvoir maîtriser le taux d'erreur (cf. section 4.3). Au final, nous résumons les différents paramètres de réglages de l'algorithme dans la section 4.4 puis nous présentons dans la section 4.5 les performances obtenues avec l'algorithme présenté dans la section 4.1.2 suivant les deux règles de décision et nous comparons celles-ci à celles obtenues avec l'algorithme des KPPV.

4.1 Modèle de mélange pour les applications NCTR

En rappellant que la base d'apprentissage est noté $\boldsymbol{X}_A$, pour chaque élément $\boldsymbol{x}_{A,i}$ de $\boldsymbol{X}_A$, on définit un modèle de mélange de la manière suivante :

$$p(\boldsymbol{x}_{A,i}|\boldsymbol{X}_A) = \sum_{k=1}^{K} \pi_A^k p(\boldsymbol{x}_{A,i}|\theta_A^k) \tag{4.1}$$

où θ_A^k est l'ensemble des paramètres de la distribution k et π_A^k est la probabilité *a priori* de la classe k.

De même, pour chaque élément de la base de test $\boldsymbol{x}_{T,i}$, on définit le modèle de mélange de la manière suivante :

$$p(\boldsymbol{x}_{T,i}|\boldsymbol{X}_A) = \sum_{k=1}^{K} \pi_A^k p(\boldsymbol{x}_{T,i}|\theta_A^k) \tag{4.2}$$

C'est cette équation qui nous intéresse car notre problème consiste bien à reconnaître un profil distance de test ($\boldsymbol{x}_{T,i}$) à partir d'une base d'apprentissage définie par les paramètres θ_A^k et π_A^k.

4.1.1 Gaussian Mixture Models (GMM)

Dans le cas d'un modèle de mélange de Gausssiennes, $\theta_A^k = \{\boldsymbol{m}_A^k, \boldsymbol{C}_A^k\}$, où $\boldsymbol{m}_A^k$ est le profil distance moyen de la classe k :

$$\boldsymbol{m}_A^k = \frac{1}{N_A^k} \sum_{i=1}^{N_A^k} \boldsymbol{x}_{A,i}^k \tag{4.3}$$

et $\boldsymbol{C}_A^k$ est la matrice de covariance de la classe k. Les élements de cette matrice sont définis de la manière suivante :

$$\boldsymbol{C}_A^k(p,q) = \frac{1}{N_A^k - 1} \sum_{i=1}^{N_A^k} (x_{A,i}^k(p) - m_A^k(p))\ (x_{A,i}^k(q) - m_A^k(q)) \tag{4.4}$$

La vraisemblance $p(\boldsymbol{x}_{T,i}|\theta_A^k)$ s'exprime comme suit :

$$p(\boldsymbol{x}_{T,i}|\theta_A^k) \quad = \quad \mathcal{N}(\boldsymbol{x}_{T,i}|\theta_A^k)$$

Finalement, on utilise le théorème de Bayes pour calculer la probabilité *a posteriori* $p(\boldsymbol{x}_{T,i} \in k|\boldsymbol{x}_{T,i})$ de chaque classe k.

$$p(\boldsymbol{x}_{T,i} \in k|\boldsymbol{x}_{T,i}) = \frac{\pi_A^k\ p(\boldsymbol{x}_{T,i}|\theta_A^k)}{\sum_i^K \pi_A^k\ p(\boldsymbol{x}_{T,i}|\theta_A^k)} \tag{4.5}$$

4.1.2 Modèle de mélange de Student

Comme on a pu le mentionner dans le chapitre 1, les distributions gaussiennnes multivariées sont de bonnes candidates pour modéliser les profils distance (Jacobs 1997, Du et al. 2008) mais elles sont sensibles aux observations aberrantes. Les distributions de Student (distribution à queues lourdes) offrent une alternative aux distributions gaussiennnes et apportent une plus grande robustesse à ces points aberrants en permettant de prendre en compte dans les paramètres du modèle des incertitudes sur les données.

On peut définir la distribution de Student comme un mélange infini de Gaussiennes ayant la même moyenne et matrice de covariance (Liu and Rubin 1995) :

$$t(\boldsymbol{x}_{T,i}|\boldsymbol{m}_A^k, \boldsymbol{C}_A^k, \nu) = \int_0^\infty \mathcal{N}\left(\boldsymbol{x}_{T,i} \mid \boldsymbol{m}_A^k, \frac{\boldsymbol{C}_A^k}{u}\right) \mathcal{G}\left(u \mid \frac{\nu}{2}, \frac{\nu}{2}\right) du \tag{4.6}$$

avec $\mathcal{N}\left(\boldsymbol{x}_{T,i}|\boldsymbol{m}_A^k, \boldsymbol{C}_A^k\right)$ la loi normale définie à partir des paramètres introduits dans la section 4.1.1 et u qui suit une loi Gamma dépendant uniquement du degré de liberté ν.

Cette définition est intéressante car elle permet de faire le lien avec les mélanges de distributions gaussiennnes.

Comme avec un classique algorithme GMM, quand un profil distance sous test $\boldsymbol{x}_{T,i}$ est mesuré, on applique la règle de Bayes (cf. équation 4.7) pour calculer la probabilité *a posteriori* de chaque classe k :

$$p(\boldsymbol{x}_{T,i} \in k|\boldsymbol{x}_{T,i}) = \frac{\pi_A^k\ p(\boldsymbol{x}_{T,i}|\theta_A^k, u)}{\sum_i^K \pi_A^k\ p(\boldsymbol{x}_{T,i}|\theta_A^k, u)} \tag{4.7}$$

avec :

$$p(\boldsymbol{x}_{T,i}|\theta_A^k, u) = \mathcal{N}(\boldsymbol{x}_{T,i}|\boldsymbol{m}_A^k, \frac{\boldsymbol{C}_A^k}{u}) \tag{4.8}$$

En effet, en reprenant 4.6 on a bien :

$$\begin{aligned} t(\boldsymbol{x}_{T,i}|\boldsymbol{m}_A^k, \boldsymbol{C}_A^k, \nu) &= \int_0^\infty \mathcal{N}\left(\boldsymbol{x}_{T,i} \mid \boldsymbol{m}_A^k, \frac{\boldsymbol{C}_A^k}{u}\right) \mathcal{G}\left(u \mid \frac{\nu}{2}, \frac{\nu}{2}\right) du \\ &= \int_0^\infty p(\boldsymbol{x}_{T,i}|\theta_A^k, u)\ p(u)\ du \end{aligned}$$

4.2 Estimation des paramètres et calcul des probabilités *a posteriori*

Dans cette section, nous nous intéressons dans un premier temps à la manière utilisée pour estimer les paramètres du modèle de mélange. Dans un second temps, nous abordons le problème du calcul des probabilités *a posteriori*.

4.2.1 Estimation des paramètres

Notre problème est un problème supervisé (c'est-à-dire que nous connaissons les classes des profils distance de la base d'apprentissage), il n'y a donc pas de problème pour estimer les paramètres du modèle. Les paramètres $\boldsymbol{m}_A^k$, $\boldsymbol{C}_A^k$ et π_A^k sont donc estimés en utilisant la base d'apprentissage. Néanmoins, il faut tout de même prendre en compte dans l'estimation des paramètres $\boldsymbol{m}_A^k$ and $\boldsymbol{C}_A^k$ l'étape de seuillage des profils distance et l'étape de recalage des profils distance d'apprentissage par rapport au profil distance sous test. En effet, dans la pratique dès qu'un profil distance est mesuré (profil distance de la base de test), on cherche à le classifier. On ne peut donc pas calculer à l'avance les seuils pour l'étape de seuillage car on ne connaît pas le niveau de bruit des profils distance de test à l'avance. Cela veut dire qu'à chaque nouveau profil distance de test, il faut recalculer les seuils et réestimer les paramètres du mélange. D'autant plus que pour chaque nouveau profil distance de test, les profils distance d'apprentissage sont récalés par rapport à celui-ci. L'étape de recalage nous oblige donc également à réestimer pour chaque profil distance de test les paramètres du modèle de mélange. En terme de coût de calcul, le nombre d'opérations flottantes nécessaires ne sera donc pas inférieur à celui des KPPV (cf. section 6.4).

Le paramètre ν est quant à lui estimé en utilisant la connaissance que nous avons sur la fluctuation de la SER dans une case distance. On fait l'hypothèse que la mesure de la SER dans une case distance pour un profil distance en linéaire suit une loi lognormale centrée sur la valeur de la SER et de variance dépendante de la valeur de la SER et d'un paramètre k fixé de manière empirique. A partir de cette hypothèse, on en déduit que la mesure de la SER dans une case distance pour un profil distance en dB suit une loi normale $\mathcal{N}(m, \sigma_n^2)$. La valeur de m et de σ_n^2 est déduite des paramètres estimés pour la loi lognormale. Le paramètre qui nous intéresse est le paramètre σ_n^2. Ce qu'il est intéressant de noter c'est qu'avec des profils distance exprimés en dB, le paramètre σ_n^2 ne dépend pas de la

valeur de la SER mais uniquement du paramètre k. Ce terme σ_n^2 peut donc être calculé en amont et ne dépend pas de la base d'apprentissage.

Pour pouvoir intégrer cette variance sur la mesure de la SER dans notre modèle, nous allons paramétrer la loi Gamma de notre modèle (cf. équation 4.6) de manière à ce que la variance de cette loi soit égale au paramètre σ_n^2. Une loi Gamma est paramétrée par deux paramètres α et β qui sont pour notre modèle tous les deux égaux à $\frac{\nu}{2}$. L'espérance d'une loi Gamma est égale à $\frac{\alpha}{\beta}$ et la variance à $\frac{\alpha}{\beta^2}$. Vu que $\alpha = \beta$, on peut calculer la valeur du paramètre ν qui permet d'avoir la variance de la loi Gamma égale à σ_n^2.

4.2.2 Calcul des probabilités *a posteriori*

Le principal problème de ce type de modèle réside surtout dans le calcul des probabilités *a posteriori* car cela nécessite d'inverser des matrices de covariance $\boldsymbol{C}_T^k$ le plus souvent très mal-conditionnées. En effet, les profils distance haute résolution ont une résolution inférieure au mètre, ce qui permet d'observer de très petites fluctuations de SER le long d'une cible. Le revers de la médaille est qu'avec les méthodes de type probabiliste, se pose le problème du fléau de la dimension. Ce problème vient du fait que dans un espace de grande dimension, il existe des directions où l'information est, si ce n'est nulle, très proche de zéro, ce qui rend les matrices de covariance singulières et donc difficilement inversibles.

Ce problème est très souvent abordé dans la littérature. Pour résumer, les solutions actuelles qui permettent de palier à ce problème consistent soit à réduire la dimension des données, soit à régulariser les estimations des matrices de covariance ou soit à utiliser une modélisation des données qui soit parcimonieuse. Les travaux de Charles Bouveyron (Bouveyron 2006) sur le sujet résument très bien la problèmatique de l'estimation de paramètres dans un espace de grande dimension et illustrent très bien les limites de la réduction de dimension, de la régularisation ou de l'utilisation des représentations parcimonieuses pour régler ce problème. Bouveyron propose donc une reparamétrisation du modèle de mélange Gaussien qui permet de combiner les idées de réduction de dimension, de contraintes sur le modèle et de régularisation. Cette reparamétrisation s'inspire de l'approche de Celeux et Govaert (Celeux and Govaert 1993), basée sur la décomposition spectrale de la matrice de covariance $\boldsymbol{C}_A^k$ (Bandfield and Raftery 1993).

$$\boldsymbol{C}_A^k = Q^k \Delta^k (Q^k)^T \tag{4.9}$$

où Q^k est la matrice orthogonale de taille $N \times N$ contenant les vecteurs propres de $\boldsymbol{C}_A^k$ et Δ^k est la matrice de covariance de la classe k dans son espace propre. La matrice Δ^k est alors une matrice diagonale contenant les valeurs propres de $\boldsymbol{C}_A^k$.

Cette reparamétrisation est utilisée de la même manière avec le modèle de mélange de distributions de Student, mais au lieu de reparamétriser $\boldsymbol{C}_A^k$, on reparamétrise $\frac{\boldsymbol{C}_A^k}{u}$.

Bouveyron propose de modéliser le fait que les données de chaque classe vivent dans des sous-espaces de dimensions inférieures à la dimen-

sion de l'espace et donc d'écrire la matrice diagonale Δ^k sous la forme suivante :

$$\Delta^k = \begin{pmatrix} \begin{bmatrix} a_{k,1} & & 0 \\ & \ddots & \\ 0 & & a_{k,d_k} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} b_k & & 0 \\ & \ddots & \\ & & \ddots \\ 0 & & b_k \end{bmatrix} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} d_k \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} (d - d_k) \end{matrix} \tag{4.10}$$

où $a_{k,j} >= b_k$, $j = 1, \ldots, d_k$ et $d_k < d$ pour tout $k = 1, \ldots, K$. Sous cette forme, on considère donc que les d_k plus grandes valeurs propres de chaque combinaison k, sont distinctes et que les $(d - d_k)$ plus petites valeurs propres sont égales.

Ce modèle nécessite donc uniquement l'estimation de K sous-espaces de dimensions $d_1, \ldots, d_K$. Ainsi plus les dimensions d_k sont petites comparées à la dimension de l'espace, plus le modèle est parcimonieux.

Au final, ce modèle ne nécessite pas l'inversion des matrices de covariance $\boldsymbol{C}_A^k$ car les termes de variance sont résumés dans les paramètres a_{kj} et b_k. Le calcul des probabilités *a posteriori* se résume aux calculs des fonctions de coût associées à chaque classe k. Elles sont définies de la manière suivante (cf. thèse de Bouveyron (Bouveyron 2006)) :

$$\begin{aligned} R_k(\boldsymbol{x}_{T,i}) &= ||\boldsymbol{m}_A^k - H_k(\boldsymbol{x}_{T,i})||_{A_k}^2 + \frac{1}{b_k}||\boldsymbol{x}_{T,i} - H_k(\boldsymbol{x}_{T,i})||^2 + \sum_{j=1}^{d_k} \log(a_{kj}) \\ &\quad + (d - d_k)\log(b_k) - 2\log(\pi_A^k) + d\log(2\pi) \end{aligned}$$

avec $||.||_{A_k}$ une norme sur $\mathbb{E}_k$, le sous-espace engendré par les d_k vecteurs propres associés aux valeurs propres $a_{k1}, \ldots, a_{kd_k}$ et H_k l'opérateur de projection sur le sous-espace $\mathbb{E}_k$.

La probabilité *a posteriori* est ensuite déduite de R_k :

$$P_k = \frac{\exp(-\frac{1}{2}R_k)}{\sum_{k=1}^{K} \exp(-\frac{1}{2}R_k)} \tag{4.11}$$

4.3 Règles de décision

4.3.1 Règle de décision du MAP

La règle de décision du MAP est la règle de décision la plus souvent rencontrée dans les problèmes de classification. Le principe est assez simple : une fois les probabilités *a posteriori* de chaque classe calculées,

la classe ayant la probabilité la plus importante est la classe retenue. Autrement dit, soit $\boldsymbol{p} = [p_1, \ldots, p_K]$, le vecteur des probabilités *a posteriori* calculées pour les K classes.

La classe retenue est donc la classe k avec :

$$k = \arg\max_k(\boldsymbol{p}) \tag{4.12}$$

Le problème avec cette règle de décision est qu'elle n'interdit pas de retenir une classe avec une probabilité de 40% par exemple. Dans notre optique de maîtrise du taux d'erreur, cela n'est pas acceptable.

4.3.2 Règle de décision pour le contrôle du taux d'erreur

Une manière de maîtriser le taux d'erreur est de ne prendre une décision uniquement lorsque la confiance quant à l'appartenance à une classe est suffisamment forte, c'est-à-dire lorsque la probabilité est suffisamment élevée.

Reprenons le vecteur des probabilités *a posteriori* $\boldsymbol{p} = [p_1, \ldots, p_k]$. La règle de décision modifiée est la suivante :

- si $p_k \geq SA$, la classe k est acceptée
- si $SR \leq p_k < SA$, la classe k est incertaine
- si $p_k < SR$, la classe k est rejetée

Au lieu d'avoir une règle de décision binaire (Acceptation/Rejet), on introduit un troisième niveau de décision (Acceptation/Incertitude/Rejet). Une solution pour règler les seuils de décision est d'utiliser la base d'apprentissage. L'algorithme est appliqué en amont sur les profils de la base d'apprentissage pour différentes valeurs de SR et SA. En traçant, la courbe du taux d'erreur en fonction de SR, on peut fixer la valeur de SR permettant de rester inférieur à un taux d'erreur fixé. Ensuite, une fois SR fixé, on trace la courbe du taux de succès en fonction de SA, la valeur de SA étant celle pour laquelle le taux de succès est maximal.

Avec cette règle de décision, on peut construire trois matrices de décision : (i) une matrice d'acceptation (**MA**), (ii) une matrice d'incertitude (**MI**) et (iii) une matrice de rejet (**MR**). Chacune de ces matrices est une matrice de taille $K \times K$.

La diagonale de **MR** donne le taux d'erreur (TE) pour chaque classe :

$$TE(i) = MR(i, i) \tag{4.13}$$

La règle de décision définie dans cette section est la règle de décision que nous avons utilisée pour évaluer les performances.

4.4 Paramètres de réglages

A partir de la description faite dans les sections précédentes, il apparaît que plusieurs paramètres peuvent influencer les performances de l'algorithme basé sur le mélange de distributions de Student :

- la nature des signatures et les prétraitements : comme avec les KPPV, pour réduire l'influence du bruit, les profils distance d'apprentissage et de test sont seuillés avec le même seuil qui dépend de la puissance de bruit calculée sur le profil de test et le profil d'apprentissage. Une étape de recalage est également nécessaire pour recaler les profils distance d'apprentissage au profil distance sous test.
- les paramètres du mélange de distributions de Student : les paramètres $\boldsymbol{m}_A^k$ et $\boldsymbol{C}_A^k$ dépendent de la base d'apprentissage et de la manière dont les profils distance de cette base ont été seuillés et recalés. Le paramètre ν est quant à lui fixé à partir de connaissances d'experts sur la fluctuation de la SER dans une case distance.
- les paramètres de la reparamétrisation : cette reparamétrisation permet de contrôler les caractéristiques de la k-ème composante du mélange gaussien grâce à quatres types de paramètres : le vecteur $(a_{k1}, \ldots, a_{kd_k})$, le scalaire b_k, la matrice Q_k et la dimension d_k. Les trois premiers paramètres sont calculés à partir de la décomposition spectrale de $\boldsymbol{C}_A^k$. En revanche la dimension intrasèque d_k du sous-espace de la k-ème classe est à estimer. Pour ce paramètre, nous avons utilisé, comme le préconise Bouveyron dans ces travaux, le critère empirique du scree-test de Cattell (Cattell 1966). Ce critère est très souvent utilisé dans la pratique. Il est basé sur l'analyse des différences entre les valeurs propres consécutives et permet de détecter un "coude" dans l'éboulis des valeurs propres. La dimension sélectionnée par la méthode est celle pour laquelle les différences entre les valeurs propres suivantes sont toutes plus petites qu'un certain seuil. Dans notre cas supervisé, il est possible d'estimer ce seuil par validation croisée sur la base d'apprentissage puisque l'on connaît les appartenances aux classes des profils distance de la base d'apprentissage. La valeur du seuil est celle pour laquelle le taux d'erreur calculé par validation croisée sur la base d'apprentissage est le meilleur.

4.5 Performances de l'algorithme basé sur les méthodes probabilistes

Nous présentons les performances sous la forme des trois matrices de décision définies dans la section 4.3.2 et pour les trois niveaux de RSB de notre base de données.

		MA			MI			MR		
		1	2	3	1	2	3	1	2	3
RSB=30dB	1	100	0	0	0	0	0	0	100	100
	2	0	100	0	0	0	0	100	0	100
	3	0	0	100	0	0	0	100	100	0
RSB=20dB	1	92	6	2	0	0	2	8	94	96
	2	6	92	2	0	0	0	94	8	98
	3	2	0	98	0	0	0	92	100	2
RSB=15dB	1	58	16	24	10	4	4	32	80	72
	2	6	86	6	2	0	2	92	14	92
	3	8	2	80	14	2	12	76	96	8

TAB. 4.1 : Matrices de décision **MA**, **MI** and **MR** pour trois RSB différents dans l'espace de grande dimension - Algorithme bayésien

Avec un $RSB = 30dB$, on retrouve un taux d'erreur nul et un taux de bonne identification (et un taux de succès) égale à 100% comme avec l'algorithme des KPPV. Avec un $RSB = 20dB$, il y a de plus en plus de confusions entre la classe 1 et 2 mais la classe 3 reste plutôt bien classifiée. Par contre le taux d'erreur commence à devenir un peu plus critique avec 8% d'erreur pour les classes 1 et 2. Lorsque le $RSB = 15dB$, les performances se dégradent très nettement et deviennent moins bonnes que celles obtenues avec l'algorithme des KPPV. Le taux d'erreur pour les classes 1 et 2 devient beaucoup trop important et les erreurs sont également plus nombreuses pour la classe 3.

	TE	TBI	TS
RSB=30dB	0	70.67	100
RSB=20dB	6	91.3	94.0
RSB=15dB	18	70.7	74.7

TAB. 4.2 : Taux d'erreur, de bonne identification et de succès - Algorithme bayésien

Conclusion sur les performances de l'algorithme de reconnaissance de cibles basés sur les méthodes probabilistes

Malgré la prise en compte des incertitudes dans les paramètres du modèle (via les lois de Student) et malgré l'introduction d'une nouvelle règle de décision, on s'aperçoit que cet algorithme ne permet toujours pas de contrôler le taux d'erreur lorsque le RSB diminue même si cela doit se faire au détriment du taux de bonne identification ou du taux de succès. Au final, même si les performances paraissent un peu inférieures à celles obtenues avec l'algorithme des KPPV, on retiendra avant tout que le taux d'erreur devient beaucoup trop important lorsque le RSB diminue avec cet algorithme. Cette solution ne fournit donc pas des résultats qui permettraient d'envisager son utilisation dans le contexte NCTR. Avant de

proposer d'autres algorithmes et même si le modèle n'est pas forcément le mieux adapté, il nous a paru tout de même intéressant d'étudier la possibilité de visualiser les données dans un espace 2D en se basant sur le modèle proposé. C'est l'objet de la section suivante.

4.6 Représentation graphique des données via les méthodes probabilistes

Nous avons présenté dans le chapitre 2, une méthode pour visualiser les données à partir des matrices de covariance de chaque classe. Dans cette section, nous avons utilisé le modèle défini un peu plus tôt dans le chapitre et basé sur les mélanges de distributions de Student pour proposer une autre méthode de visualisation des données plus fine que celle présentée dans le chapitre 2. Cette méthode consiste à utiliser une technique de réduction de dimension non linéaire pour obtenir une carte des profils distance dans un espace 2D puis d'appliquer notre algorithme basé sur les mélanges de distributions de Student pour visualiser les zones d'acceptation, d'incertitude et de rejet associées à notre base d'apprentissage.

Un grand nombre de techniques de réduction non linéaire de dimension ayant pour but de préserver les structures locales des données ont été présentées ces dernières années et beaucoup d'entre elles ont été passées en revue par Lee et Verleysen (Lee and Verleysen 2007). On peut mentionner les plus populaires d'entre elles : Sammon Mapping (Sammon 1969), Stochastic Neighbor Embedding (SNE) (Hinton and Roweis 2002), Isomap (Tenenbaum et al. 2000), Maximum Variance Unfolding (MVU) (Weinberger et al. 2004), Locally Linear Embedding (LLE) (Roweis and Saul 2000) et Laplacian Eigenmaps (Belkin and Nigoyi 2002). Sur des données synthétiques, ces techniques ont de bonnes performances mais sont souvent peu satisfaisantes pour visualiser des données réelles de grande dimension. En effet, la plupart de ces techniques ne sont pas capables de prendre en compte à la fois les structures globales et locales des données sur une même carte.

En 2009, van der Maaten et Hinton (van der Maaten and Hinton 2008) ont proposé une nouvelle méthode appelée t-SNE (pour t-Distributed Stochastic Neighbor Embedding en anglais) qui est capable de très bien prendre en compte les structures locales de données de grande dimension, tout en conservant leurs structures globales telles que des regroupements parmi les données à différentes échelles. La méthode t-SNE est une extension directe de la méthode SNE (pour Stochastic Neighbor Embedding en anglais).

4.6.1 Stochastic Neighbor Embedding (SNE)

Le principe de la méthode SNE est de convertir des distances euclidiennes calculées dans l'espace de grande dimension en probabilités conditionnelles représentant la similarité. La similarité entre un profil distance $\boldsymbol{x}_{T,i}$ et un profil distance $\boldsymbol{x}_{T,j}$ peut être vue comme la probabilité conditionnelle $p_{j|i}$, sachant que $\boldsymbol{x}_{T,j}$ est un voisin de $\boldsymbol{x}_{T,i}$ si les voisins sont choisis en fonction de leur densité de possibilité sous une Gaussienne cen-

trée en $\boldsymbol{x}_{T,i}$. Mathématiquement, la probabilité conditionnelle est donnée par l'équation suivante :

$$p_{j|i} = \frac{\exp\left(-||\boldsymbol{x}_{T,i} - \boldsymbol{x}_{T,j}||_2^2/(2\sigma_i^2)\right)}{\sum_{k\neq i}\exp\left(-||\boldsymbol{x}_{T,i} - \boldsymbol{x}_{T,k}||_2^2/(2\sigma_i^2)\right)} \tag{4.14}$$

où σ_i est la variance de la Gaussienne centrée en $\boldsymbol{x}_{T,i}$.

Dans l'espace de dimension réduite, pour les profils distance $\boldsymbol{y}_{T,i}$ et $\boldsymbol{y}_{T,j}$ correspondant aux profils distance $\boldsymbol{x}_{T,i}$ et $\boldsymbol{x}_{T,j}$, il est possible de calculer une probabilité conditionnelle similaire, que l'on note $q_{j|i}$

$$q_{j|i} = \frac{\exp\left(-||\boldsymbol{y}_{T,i} - \boldsymbol{y}_{T,j}||_2^2\right)}{\sum_{k\neq i}\exp\left(-||\boldsymbol{y}_{T,i} - \boldsymbol{y}_{T,k}||_2^2\right)} \tag{4.15}$$

Si les points $\boldsymbol{y}_{T,i}$ et $\boldsymbol{y}_{T,j}$ modélisent bien la similarité entre les profils distance de grande dimension $\boldsymbol{x}_{T,i}$ and $\boldsymbol{x}_{T,j}$, les probabilités conditionnelles $p_{j|i}$ et $q_{j|i}$ seront égales. La méthode SNE a donc pour but de trouver une représentation en dimension réduite (carte) qui minimise les disparités entre $p_{j|i}$ et $q_{j|i}$. La distance de Kullback-Leiber est une mesure naturelle de la fidélité avec laquelle $q_{j|i}$ modélise $p_{j|i}$. La méthode SNE minimise la somme des divergences de Kullback-Leiber sur tous les points en utilisant une méthode de "descente de gradient". La fonction de coût C associée est donnée par

$$C = \sum_i KL(P_i||Q_i) = \sum_i\sum_j p_{j|i} log\frac{p_{j|i}}{q_{j|i}} \tag{4.16}$$

où P_i représente la distribution de probabilité conditionnelle sur tous les points autres que $\boldsymbol{x}_{T,i}$ et Q_i représente la distribution de probabilité conditionnelle sur tous les points de la carte autres que $\boldsymbol{y}_{T,i}$.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

La méthode t-SNE apporte deux innovations majeures par rapport à la méthode classique SNE. Premièrement, elle utilise une version symétrique de la fonction de coût du SNE. En effet, pour minimiser la somme des divergences de Kullback-Leiber entre les probabilités conditionnelles $p_{j|i}$ and $q_{j|i}$, il est également possible de minimiser une seule divergence de Kullback-Lieber entre la distribution jointe, P, dans l'espace de grande dimension et la distribution jointe, Q, dans l'espace de dimension réduite :

$$C = KL(P||Q) = \sum_i\sum_j p_{ij} log\frac{p_{ij}}{q_{ij}} \text{ with } p_{ii} = p_{jj} = 0 \tag{4.17}$$

Dans la méthode SNE symétrique, la similarité q_{ij} entre deux points i et j de la carte est donnée par

$$q_{ij} = \frac{\exp\left(-||\boldsymbol{y}_{T,i} - \boldsymbol{y}_{T,j}||_2^2\right)}{\sum_{k\neq l}\exp\left(-||\boldsymbol{y}_{T,k} - \boldsymbol{y}_{T,l}||_2^2\right)} \tag{4.18}$$

et dans l'espace de grande dimension

$$p_{ij} = \frac{p_{j|i} + p_{j|i}}{2N_T} \tag{4.19}$$

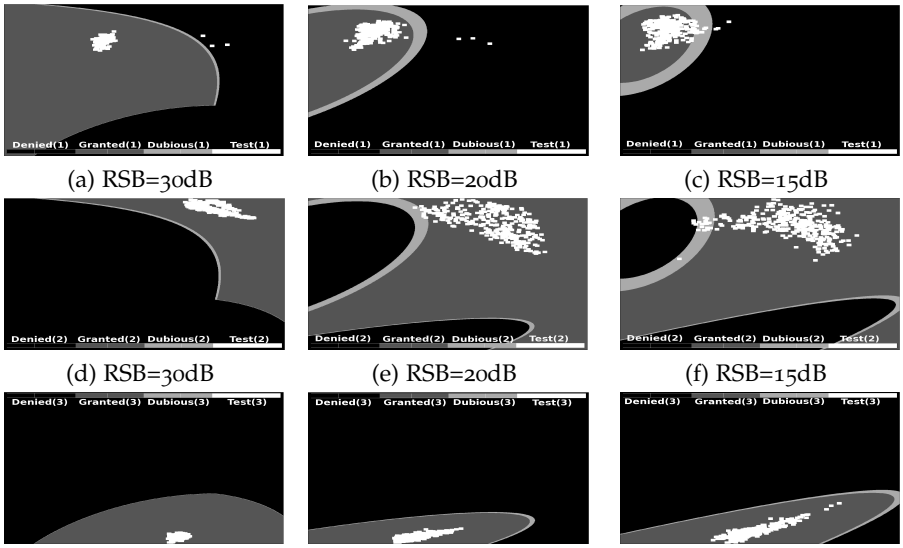
Cela assure que $\sum_j p_{ij} > \frac{1}{2N_T}$ pour tous les points $\boldsymbol{x}_{T,i}$, de manière à ce que chaque point $\boldsymbol{x}_{T,i}$ apporte une contribution significative à la fonction de coût.

Deuxièmement, la méthode t-SNE utilise une distribution de Student au lieu d'une distribution normale pour modéliser la similarité entre deux points de l'espace de dimension réduite. La méthode t-SNE utilise une distribution à queue lourde dans l'espace de dimension réduite pour atténuer à la fois le problème d'entassement des données dans certaines zones de la carte et les problèmes d'optimisation que l'on peut rencontrer avec le SNE classique. Avec le t-SNE, q_{ij} devient donc

$$q_{ij} = \frac{\left(1 + ||\boldsymbol{y}_{T,i} - \boldsymbol{y}_{T,j}||_2^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\boldsymbol{y}_{T,k} - \boldsymbol{y}_{T,l}||_2^2\right)^{-1}} \tag{4.20}$$

4.6.2 Carte de probabilité dans l'espace de dimension réduite

Après avoir calculé les coordonnées $\boldsymbol{Y}_T$ dans l'espace de dimension réduite à partir de la méthode t-SNE et des coordonnées $\boldsymbol{X}_T$ dans l'espace de grande dimension, on veut visualiser comment les données se placent par rapport à la règle de décision. En effet, à partir de $\boldsymbol{Y}_T$, les paramètres θ_T^k des K distributions de Student du modèle de mélange sont estimés. Ensuite, on peut échantillonner l'espace 2D et calculer pour chaque échantillon de la grille la probabilité *a posteriori* de chaque distribution de Student du modèle de mélange. Finalement, on peut construire une carte de probabilité où les zones d'acceptation et d'incertitude pour chaque classe sont représentées. Les profils distance de la base d'apprentissage peuvent également être placés sur cette carte de probabilité.



(a) RSB=30dB (b) RSB=20dB (c) RSB=15dB

(d) RSB=30dB (e) RSB=20dB (f) RSB=15dB

(g) RSB=30dB (h) RSB=20dB (i) RSB=15dB

FIG. 4.1 : Visualisation 2D des données à partir de la méthode t-SNE pour les classes 1, 2 and 3 (lignes) et pour les trois niveaux de RSB (colonnes).

La figure 4.1 nous permet de visualiser dans un espace 2D nos données d'apprentissage. On remarque qu'avec un RSB de $30dB$ (cf. figure 4.1a, 4.1d et 4.1g) les données d'apprentissage se situent quasiment toutes dans les zones d'acceptation (à l'exception de quelques données pour la classe 1). En revanche, plus le RSB diminue, plus le nombre d'échantillons se situant dans les zones d'incertitudes ou dans les zones de rejet augmente.

4.7 Conclusion du chapitre

Comme on a déjà pu le mentionner dans la section 4.5, l'algorithme que nous avons présenté dans ce chapitre ne remplit pas les conditions nécessaires pour être utilisé dans le contexte NCTR. Même si les performances semblent acceptables, la dégradation non controlée du taux d'erreur lorsque le RSB diminue n'est pas satisfaisante pour une application de type NCTR, où l'on doit s'assurer que le taux d'erreur reste faible quel que soit le RSB des profils distance utilisés. L'inconvénient des méthodes de type probabiliste, sous la forme présentée dans ce chapitre, est qu'elles ne permettent pas de prendre une décision indépendante classe par classe. Néanmoins, lorsque le profil distance de test appartient à l'une des classes présentes dans la base d'apprentissage, ces méthodes devraient tout de même permettre de mieux contrôler le taux d'erreur que ce qu'on a pu constater dans ce chapitre. Une première raison qui peut expliquer ces performances médiocres est le choix des distributions paramétriques utilisées pour modéliser les profils distance. Même si les distributions gaussiennes sont souvent utilisées dans le domaine, elles ne sont peut être pas suffisamment sophistiquées pour "coller" au plus près à la distribution réelle des profils distance. Cela peut donc entraîner des erreurs difficilement maîtrisables car présentes à l'origine dans le modèle. Il existe dans la littérature d'autres types de modélisation utilisant des modèles de mélange de distributions bien plus complexes qui peuvent

permettre de réduire ces erreurs mais la complexité des calculs sera également beaucoup plus importante.

Dans ce chapitre, nous avons également présenté une méthode pour visualiser les données dans un espace 2D. Cette méthode permet de construire des cartes de probabilité à partir de la base d'apprentissage et donc d'avoir un aperçu de la disposition des données selon les classes. Cette visualisation permet d'avoir une image beaucoup plus fidèle de la répartition réelle des données que celle obtenue à partir d'une ACP sur deux composantes présentée dans le chapitre 2. Contrairement à l'ACP, la technique t-SNE est une technique non linéaire qui permet de préserver les structures locales et globales des données. La méthode t-SNE est une technique qui conserve également beaucoup plus d'informations pour construire la visualisation que l'ACP à deux composantes. En effet, les mesures de similarités entre chaque profil distance dans l'espace de grande dimension sont généralement très proches de celles obtenues (par une méthode de descente de gradient) dans l'espace 2D. En revanche, on avait vu dans la section 2.3.3 que la visualisation basée sur une ACP à deux composantes ne conservait dans certains cas que 40% de l'information.

5 Algorithmes de reconnaissance de cibles basés sur la logique floue

Sommaire

5.1 Distance Floue pour la Reconnaissance Instantanée (D-FRI) . . . 84
5.1.1 Construction des fonctions d'appartenance . . . 84
5.1.2 Reconnaissance du profil distance sous test . . . 85
5.1.3 Performances . . . 89
5.2 Gabarits flous pour la Reconnaissance Instantanée (G-FRI) . . . 91
5.2.1 Etape 1 : construction de la base d'apprentissage . . . 92
5.2.2 Etape 2 : mise en forme du profil distance sous test . . . 96
5.2.3 Etape 3 : comparaison du profil distance sous test avec la base d'apprentissage . . . 97
5.2.4 Adaptation des gabarits aux autres classes de la base de données . . . 101
5.2.5 Performances . . . 101
5.3 Conclusion du chapitre . . . 103

Dans ce chapitre, nous présentons deux algorithmes NCTR basés sur la logique floue. A partir des principes de la logique floue explicités dans l'état de l'art (cf. section 1.3.2.3), nous proposons dans un premier temps, dans la section 5.1 un algorithme, nommé D-FRI (Distance Floue pour la Reconnaissance Instantanée), permettant de contrôler le taux d'erreur à partir de l'attribut "distances euclidiennes entre des profils distance" de la même classe. Ensuite, après avoir mis en avant les avantages et les inconvénients de ce premier algorithme, nous présentons une deuxième version de l'algorithme (cf. section 5.2), nommé G-FRI (Gabarits Flous pour la Reconnaissance Instantanée), permettant d'améliorer les performances observées avec la première version de l'algorithme.

5.1 Distance Floue pour la Reconnaissance Instantanée (D-FRI)

Le principe de cet algorithme est d'utiliser la distance euclidienne entre deux profils distance pour caractériser les différentes classes. Pour chaque classe, on construit une fonction d'appartenance à partir de l'attribut "distance euclidienne entre deux profils distance". On veut modéliser le fait que les distances entre des profils distance d'une même classe k ne sont pas distribuées de la même manière que les distances entre des profils distance d'une autre classe $j \neq k$. L'objectif de cet algorithme est de faire de la reconnaissance de cibles non coopératives en gardant un contrôle sur le taux d'erreur et en maximisant le taux de succès. Lorsqu'on parle de contrôle du taux d'erreur, cela signifie que l'on veut s'assurer que la probabilité de prendre la décision "la classe $\neq$ k" sachant que "la classe $=$ k" est inférieure à une certaine valeur (par exemple 5%).

L'algorithme D-FRI est composé de deux étapes majeures :
- Construction des fonctions d'appartenance pour chaque classe.
- Reconnaissance du profil distance sous test.

5.1.1 Construction des fonctions d'appartenance

Soit $\mathbf{\Theta}^k$, la matrice des distances entre les profils distance de la matrice $\boldsymbol{X}_A^k$. Chaque θ_{ij}^k est la distance entre le i-ème et j-ème profil distance de $\boldsymbol{X}_A^k$. La diagonale de $\mathbf{\Theta}^k$ est égale à 0.

$$\theta_{ij}^k = ||\boldsymbol{x}_{A,i}^k - \boldsymbol{x}_{A,j}^k||^2 \tag{5.1}$$

Soit $\mathbf{m\Theta}^k = [m\theta_1^k, \ldots, m\theta_{L_A^k}^k]$, le vecteur des minima des distances entre les profils distance de la base d'apprentissage appartenant à la classe k.

$$m\theta_i^k = \min_j(\theta_{ij}^k), \forall i \neq j \tag{5.2}$$

Pour calculer la fonction d'appartenance de chaque classe, l'histogramme de chaque $\mathbf{m\Theta}^k$ est calculé. A partir de cet histogramme, les fonctions de distribution de chaque classe (DF_k) sont déterminées. Pour chaque DF_k, on cherche à calculer la valeur du percentile 95%, c'est-à-dire que l'on cherche $m\theta_{lim}^k$ tel que $P(m\theta_i^k \leq m\theta_{lim}^k) = 0.95$. Pour chaque classe, on obtient donc une distance $m\theta_{lim}^k$ qui nous permet de construire une fonction d'appartenance pour chaque classe. En effet, si le profil distance de test appartient à la classe k, il y a 95% de chance que la distance entre ce profil distance de test et un profil distance de la base d'apprentissage de la classe k soit inférieure à $m\theta_{lim}^k$. On veut utiliser le fait que la différence entre deux profils distance de deux classes différentes est *a priori* plus importante que la différence entre deux profils distance de la même classe.

Soit M_k, la fonction d'appartenance de la classe k.

$$M_k(m\theta_i^k) = \begin{cases} 1 & \text{si } m\theta_i^k \leq m\theta_{lim}^k \\ 0 & \text{sinon.} \end{cases} \tag{5.3}$$

La figure 5.1 résume les différentes étapes pour construire les fonctions d'appartenance.

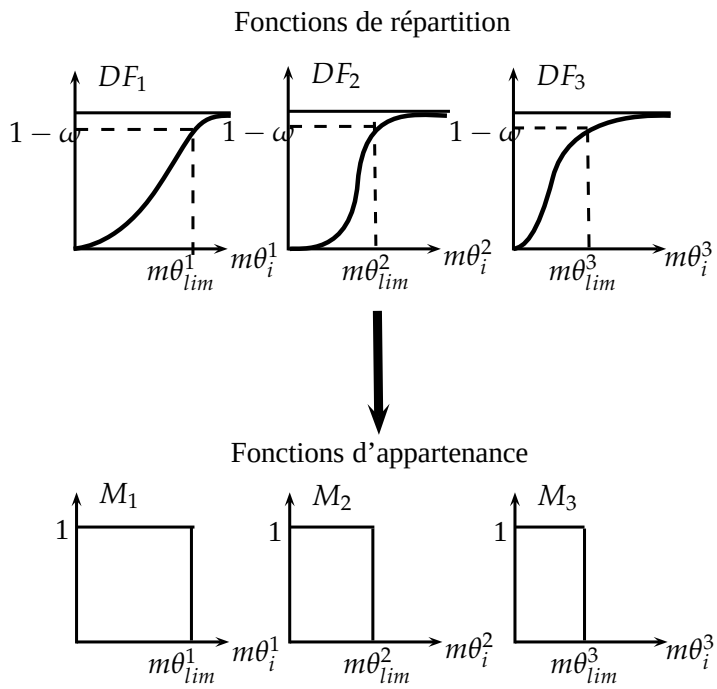


FIG. 5.1 : Construction des fonctions d'appartenance

Cette première étape de construction des fonctions d'appartenance s'effectue "hors-ligne" car les fonctions d'appartenance sont construites uniquement à partir des profils distance d'apprentissage.

5.1.2 Reconnaissance du profil distance sous test

La seconde étape de l'algorithme est l'étape de reconnaissance en temps réel. Elle démarre lorsqu'un nouveau profil distance est mesuré par le radar. Cette seconde étape est elle-même découpée en 4 sous-étapes :

- Calcul des distances entre le profil distance sous test et les profils distance d'apprentissage ;
- Calcul des densités de possibilité associées aux distances calculées à l'étape précédente ;
- Calcul des possibilités associées aux densités de possibilité ;
- Décisions.

Avant tout chose, on a besoin de définir un modèle permettant de prendre en compte l'erreur sur la mesure de distance. En effet, jusqu'à présent, le degré d'appartenance à la classe k était décrit à travers la distance entre deux profils distance. Cependant, l'erreur possible sur cette mesure n'a pas été prise en compte. Pour la prendre en compte, on utilise la notion de densité de possibilité.

La distance utilisée dans notre algorithme est la distance euclidienne. Soit $\boldsymbol{x}$ et $\boldsymbol{y}$ deux profils distance quelconques de taille N :

$$d_{euclidian} = \sum_{i}^{N} (x_i - y_i)^2 \tag{5.4}$$

On fait l'hypothèse que le bruit de la mesure de la SER dans une case distance suit une loi normale, c'est-à-dire que $x_i \sim \mathcal{N}(m, \sigma_n^2)$ et $y_i \sim \mathcal{N}(m, \sigma_n^2)$. Soit $z_i = x_i - y_i$ alors $z_i \sim \mathcal{N}(0, 2\sigma_n^2)$.

On a donc :

$$\sum_{i=1}^{N} \frac{z_i^2}{2\sigma_n^2} \sim \chi_{nc}^2(N, \lambda) \tag{5.5}$$

avec χ_{nc}^2, la loi du chi2 non-centrée et $\lambda = \sum_{i=1}^{N} \left(\frac{m_i}{\sigma_i}\right)^2$.

L'équation (5.5) peut être réécrite de la manière suivante :

$$\alpha = \frac{d_{euclidian}}{2\sigma_n^2} \sim \chi_{nc}^2(N, \lambda) \tag{5.6}$$

La variable aléatoire α suit une distribution du chi2 non centrée avec N degrés de liberté et un décentrage de λ. Il faut néanmoins tenir compte du fait que le profil distance sous test et tous les profils distance de la base d'apprentissage sont seuillés (pour réduire l'influence du bruit) par un seuil qui varie en fonction du niveau de bruit de chaque profil distance sous test. Le nombre de cases distance N utilisées pour calculer la distance varie donc d'un profil distance sous test à un autre. Pour chaque profil distance sous test, le degré de liberté de la distribution du chi2 non centrée sera donc différent.

Première étape : calcul des distances

La première étape de l'algorithme consiste à calculer les distances entre le profil distance sous test (un profil distance de X_A) et tous les profils distance de la base d'apprentissage X_A. Les distances calculées sont divisées en trois vecteurs, un pour les distances entre le profil distance sous test et les profils distance d'apprentissage appartenant à la classe 1 (X_A^1) et ainsi de suite pour les $K = 3$ classes. Ces vecteurs sont notés $\boldsymbol{d}^1$, $\boldsymbol{d}^2$ et $\boldsymbol{d}^3$. La figure 5.2 illustre en 2D cette première étape.

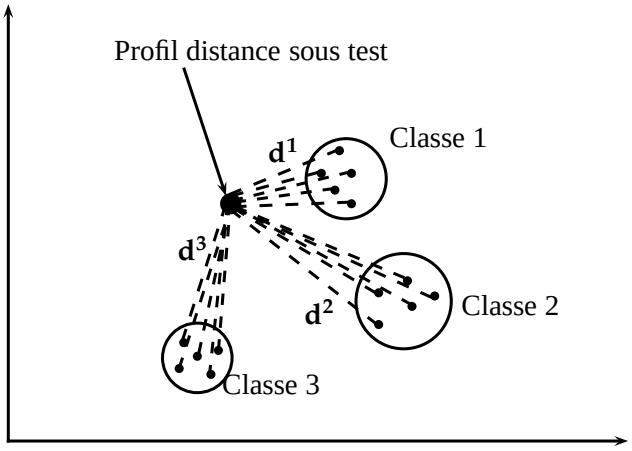


FIG. 5.2 : Calcul des distances

Deuxième étape : calcul des densités de possibilité

Dans la seconde étape, les densités de possibilités associées aux distances calculées à la première étape sont calculées. Soit d_j^k, la distance entre le profil distance sous test j et le profil distance k de la base d'apprentissage. Nous avons vu que $\alpha_j^k = d_j^k/(2\sigma_n^2)$ suit une loi du chi2 à N degrés de liberté et décentrée de λ.

La densité de possibilité associée à α_j^k peut donc être définie de la manière suivante :

$$DP_{i,j}^k = \alpha_j^k - t_i \tag{5.7}$$

avec $i \in \{1, 2\}$. t_1 et t_2, les percentiles de la distribution du chi2 décentrée permettant de traduire la répartition des erreurs de mesure. Les densités de possibilité, quant à elles, ne sont pas centrées autour de la valeur de la distance car la mesure de la distance ne peut être sous-estimée. La figure 5.3 résume cette seconde étape.

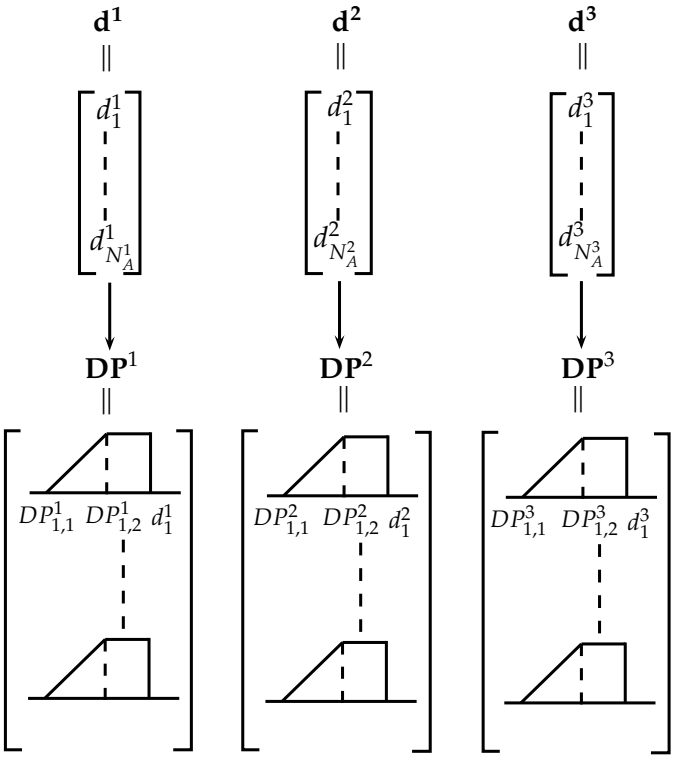


FIG. 5.3 : Calcul des densités de possibilité

Troisième étape : calcul des possibilités

Une fois les densités de possibilité calculées, chacune d'entre elles ($\mathbf{DP}^1$, $\mathbf{DP}^2$ et $\mathbf{DP}^3$) sont superposées respectivement aux fonctions d'appartenance M_1, M_2 et M_3 de manière à déterminer la possibilité associée. L'intersection de $\mathbf{DP}^1$, $\mathbf{DP}^2$ et $\mathbf{DP}^3$ avec M_1, M_2 et M_3 donne les trois vecteurs des possibilités $\boldsymbol{p}^1$, $\boldsymbol{p}^2$ et $\boldsymbol{p}^3$ (cf. figure 5.4).

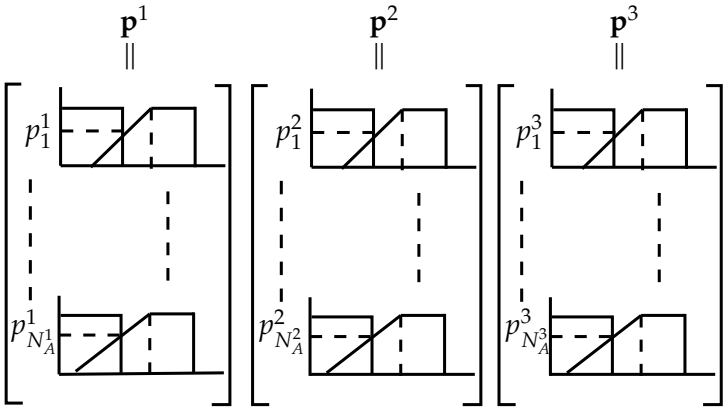


FIG. 5.4 : Calcul des possibilités

Quatrième étape : prise de décision

Une décision est prise à partir des valeurs des possibilités calculées à l'étape précédente. Tout d'abord, on fusionne l'ensemble des possibilités

calculée pour chaque classe ($\boldsymbol{p}^1$, $\boldsymbol{p}^2$, $\boldsymbol{p}^3$) pour obtenir une seule valeur de possibilité par classe.

$$mP^k = \max(\boldsymbol{p}^k) \tag{5.8}$$

Ensuite, la décision est prise en fonction de ces possibilités fusionnées :
- La classe k est acceptée pour le profil distance sous test, si $mP^k \geq SA$ (**Acceptation**).
- La classe k est déclarée incertaine pour le profil distance sous test, si $SR \leq mP^k \leq SA$ (**Incertitude**).
- La classe k est rejetée pour le profil distance sous test, si $mP^k \leq SR$ (**Rejet**).

où SA et SR sont ajustés de manière à obtenir le taux désiré. [1]

Avec cet algorithme, plusieurs classes peuvent être acceptées, déclarées incertaines ou rejetées pour un profil distance sous test. La décision est prise indépendemment pour chaque classe. La plupart des algorithmes prennent une seule décision pour chaque profil distance sous test même si la probabilité d'appartenir à la classe est faible. Avec cet algorithme, on préfère retenir plusieurs classes lorsque l'indécision est trop forte.

On contruit à nouveau trois matrices de décision à partir de la règle de décision définie ci-dessus : (i) une matrice d'acceptation (**MA**), (ii) une matrice d'incertitude (**MI**) et (iii) une matrice de rejet (**MR**).

5.1.3 Performances

Nous présentons les performances de l'algorithme pour deux jeux de données. Le premier est le jeu de données synthétiques que nous avons utilisé jusqu'à présent avec trois niveaux de RSB différents. Puis nous confrontons notre algorithme à des données réelles.

[1] L'utilisation d'une fonction d'appartenance rectangulaire permet de définir plus facilement SA et SR.

Données synthétiques

		MA			MI			MR		
		1	2	3	1	2	3	1	2	3
RSB=30dB	1	100	0	0	0	2	2	0	98	98
	2	8	100	2	4	0	0	88	0	98
	3	0	0	100	0	0	0	100	100	0
RSB=20dB	1	100	64	68	0	2	0	0	34	32
	2	82	100	68	2	0	8	16	0	24
	3	4	2	100	4	0	0	92	98	0
RSB=15dB	1	100	90	90	0	0	0	0	10	10
	2	100	100	100	0	0	0	0	0	0
	3	58	28	100	4	2	0	38	70	0

TAB. 5.1 : Matrices de décision **MA**, **MI** et **MR** pour trois RSB différents - Algorithme D-FRI

Contrairement aux deux précédents algorithmes, lorsque le RSB diminue, le taux d'erreur reste à un niveau stable. Ce résultat est intéressant car c'est vraiment le comportement recherché pour une application NCTR. En contrepartie, on se rend compte que l'on a tendance à retenir dans de nombreux cas plus d'une classe par décision, c'est-à-dire que lorsque deux classes sont possibles, on préfère accepter les deux classes.

	TE	TBI	TS
RSB=30dB	0	96.0	97.3
RSB=20dB	0	42.6	44.6
RSB=15dB	0	16.0	17.3

TAB. 5.2 : Taux d'erreur, de bonne identification et de succès - Algorithme D-FRI

Le tableau 5.2 confirme que le taux d'erreur est bien contrôlé. Le taux de succès est élevé lorsque le RSB est important. Il décroit fortement lorsque le RSB diminue. Cela est dû en parti à l'attribut distance utilisé pour calculer les fonctions d'appartenance et les densités de possiblité. Les distances entre des profils distance de la même classe auront tendance à être très faibles. Plus le RSB est faible et plus le nombre d'échantillons sur lequel est calculé la distance est faible et plus les classes sont difficiles à distinguer entre elles. On a également testé cet algorithme sur des données réelles. Les résultats sont présentés dans la section suivante.

Données réelles

Les performances de l'algorithme sont évaluées en terme de taux d'erreur (TE), taux de succès (TS) et taux de bonne identification (TBI). Le

tableau 5.3 résume les performances de l'algorithme D-FRI avec les données réelles.

		MA			**MI**			**MR**		
		1	2	3	1	2	3	1	2	3
Données réelles	1	100	100	64	0	0	15	0	0	21
	2	100	99	93	0	0	6	0	1	1
	3	100	94	100	0	4	0	0	2	0

Tab. 5.3 : Matrices de décision **MA**, **MI** et **MR** pour les données réelles - Algorithme D-FRI

On peut calculer les taux d'erreur, taux de succès et taux de bonne identification pour l'algorithme D-FRI. Les résultats sont données dans le tableau 5.4

	TE	TBI	TS
Données réelles	0	0	0

Tab. 5.4 : Taux d'erreur, taux de succès et taux de bonne identification - Algorithme D-FRI

Avec cet algorithme, on a bien un contrôle du taux d'erreur mais le taux de succès reste nul pour chaque classe car la bonne classe n'est jamais retenue seule. En regardant les matrices d'acceptation, d'incertitude et de rejet, on se rend compte que les profils distance de test des classes 2 et 3 sont pratiquement toujours retenus comme appartenant à la classe 1. Pour les profils distance de la classe 1, on arrive tout de même à rejeter dans 21.3% des cas la classe 3 mais la classe 2 est retenue dans 100% des cas.

Les performances observées avec cet algorithme nous ont poussé à développer un autre algorithme afin d'améliorer les performances en terme de taux de succès tout en gardant comme objectif la maîtrise du taux d'erreur. C'est l'objet de la section suivante.

5.2 Gabarits flous pour la Reconnaissance Instantanée (G-FRI)

A la vue des résultats obtenus avec l'algorithme D-FRI sur les données réelles, nous nous sommes concentrés sur l'objectif de définir un algorithme permettant d'améliorer les performances en terme de taux de bonne identification et de succès pour les données réelles. Les données que nous utilisons dans cette section sont donc uniquement des données réelles. Dans ce nouvel algorithme, on ne caractérise plus directement l'appartenance d'un profil distance à une classe. Le principe de ce nouvel algorithme est de travailler case distance par case distance. On va donc chercher à savoir si la valeur du signal case distance par case distance est compatible avec l'appartenance à la classe k. On construira donc une fonction

d'appartenance et une densité de possibilité pour chaque case distance et en fonction de la valeur de la possibilité dans chaque case distance ou plus précisément de la proportion de cases distance avec une possibilité suffisamment importante, on prendra une décision quant à l'appartenance du profil distance de test à la classe k. Une autre différence avec l'algorithme présenté dans la section précédente est la prise en compte de l'angle de gisement de la cible et de sa précision lors de la construction des fonctions d'appartenance. En effet, avec les données synthétiques les profils distance se trouvaient tous dans une plage azimutale restreinte, or cela n'est plus le cas avec les données réelles (par contre la variation du site reste faible même avec les données réelles). Cela peut expliquer en partie les mauvaises performances observées avec l'algorithme D-FRI. On affine également la construction des fonctions d'appartenance en utilisant des données issues d'enregistrement en chambre sourde pour une des trois classes présentes dans la base de données. Nous verrons également comment transposer cette information, disponible pour une seule classe, aux autres classes de la base de données.

5.2.1 Etape 1 : construction de la base d'apprentissage

Cette première étape consiste à construire la base de données d'apprentissage qui sera utilisée pour faire la reconnaissance d'un nouveau profil distance mesuré par le radar (profil distance de la base de test). Dans l'algorithme D-FRI, nous utilisions les profils distance de la base d'apprentissage ayant des angles de gisement relativement proches pour construire les fonctions d'appartenance. Dans ce nouvel algorithme, en plus des profils distance de la base d'apprentissage, nous introduisons pour construire les fonctions d'appartenance, des profils distance mesurés en chambre sourde.

Acquisition des profils distance en chambre sourde

L'avion est placé dans la chambre sourde avec un certain angle de gisement. Pour cet angle de gisement, on calcule une série de M profils distance de la cible pour différentes positions moteur de la cible. Ces différentes positions moteur correspondent à des rotations de quelques degrés des aubes moteurs (cf. figure 5.5).

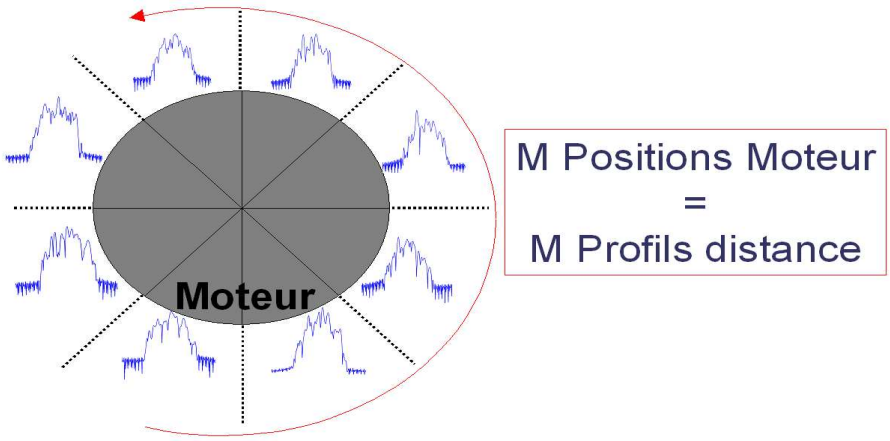


FIG. 5.5 : Positions moteur et gabarit

L'ensemble des positions moteur constitue un cycle complet du moteur. Ensuite, on effectue une rotation cette fois-ci de toute la cible de 0.05° en azimut et on recommence le même processus.

Exploitation des données en chambre sourde : construction des gabarits

Pour chaque angle de gisement par pas de 0.05°, on dispose donc d'une série de profils distance calculés pour toutes les positions des aubes moteurs. Cette série de profils distance permet donc de modéliser dans chaque case distance les fluctuations de la SER dues à cette rotation des aubes moteurs. L'idée est donc de construire un gabarit qui modélise ces fluctuations de la SER sur les profils distance mesurés en chambre sourde.

La construction d'un gabarit pour un angle de gisement donné consiste à calculer, à partir de la série des M profils distance disponibles pour un angle de gisement donné, un profil distance inférieur et un profil distance supérieur traduisant la plage de variation des valeurs de la SER dans chaque case distance (cf. figure 5.6). On note Gab_q^k, le gabarit associé à l'angle de gisement q pour la classe k.

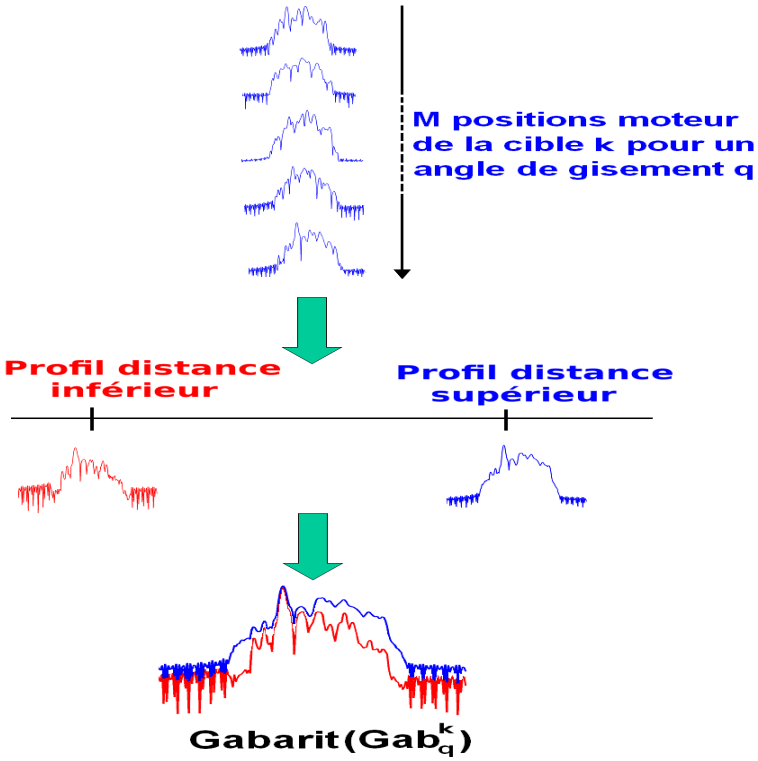


FIG. 5.6 : Positions moteur et profils distance

Construction des fonctions d'appartenance : application des gabarits aux profils distance d'apprentissage

Une fois les gabarits Gab_q^k calculées, on va chercher à les appliquer sur les profils distance de la base d'apprentissage. En effet, lorsqu'un radar mesure un profil distance p d'une cible, il est capable d'estimer son angle de gisement $g_{A,p}$ avec une certaine précision $\sigma_{g_{A,p}}$. Cette estimation est faite lors du pistage de la cible à partir de sa trajectoire et d'autres paramètres comme la vitesse de la cible. Pour chaque profil distance d'apprentissage, on peut donc dire que son angle de gisement se situe dans la plage angulaire $\mathbf{Pl}_{A,p}$ définie de la manière suivante :

$$\mathbf{Pl}_{A,p} = \left[g_{A,p} - \sigma_{g_A,p}, g_{A,p} + \sigma_{g_A,p}\right] \tag{5.9}$$

Le principe est ensuite assez simple. On applique sur le profil d'apprentissage p, les gabarits observées en chambre sourde pour chaque angle de gisement de la plage $\mathbf{Pl}_{A,p}$ de manière à construire une fonction d'appartenance pour chaque angle de gisement de cette plage. Cette fonction d'appartenance est notée $FA_{p,q}^k$ et s'exprime de la manière suivante :

$$FA_{p,q}^k = \{FA_{p,q}^{k,1}, FA_{p,q}^{k,2}\} \tag{5.10}$$

On note donc $FA_{p,q,i}^k$ la fonction d'appartenance associée à la case distance i du profil distance d'apprentissage $\boldsymbol{x}_{A,p}$ appartenant à la classe k et pour l'angle de gisement q. La figure 5.7 illustre la forme de $FA_{p,q,i}^k$

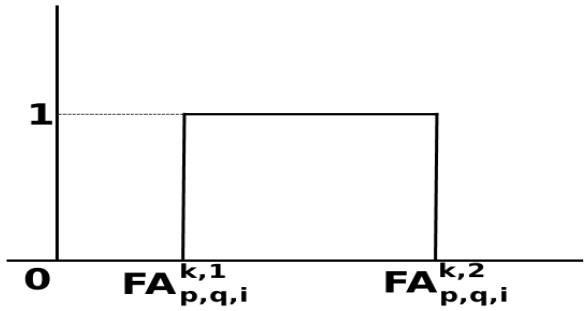


FIG. 5.7 : Fonction d'appartenance

Prenons un exemple concret afin d'illustrer le mécanisme décrit ci-dessus. On considère que notre base d'apprentissage est constituée de 3 profils distance ($\boldsymbol{x}_{A,1}$, $\boldsymbol{x}_{A,2}$ et $\boldsymbol{x}_{A,3}$). Ces profils distance ont les propriétés suivantes :

- $\boldsymbol{x}_{A,1}$ ($g_{A,1} = 90°$, $\sigma_{g_A,1} = 0.03$) $\rightarrow$ $\mathbf{Pl}_{A,1}$
- $\boldsymbol{x}_{A,2}$ ($g_{A,2} = 90.17°$, $\sigma_{g_A,2} = 0.07$) $\rightarrow$ $\mathbf{Pl}_{A,2}$
- $\boldsymbol{x}_{A,3}$ ($g_{A,3} = 90°$, $\sigma_{g_A,3} = 0.05$) $\rightarrow$ $\mathbf{Pl}_{A,3}$

A partir de ces propriétés, on construit un tableau dont les lignes sont les profils distance d'apprentissage classés par ordre de gisement croissant et les colonnes sont les valeurs des azimuts des gabarits issus des mesures en chambres sourde. Par exemple, pour la ligne 1 du tableau, on remplit les différentes colonnes à traiter en fonction des valeurs de $\mathbf{Pl}_{A,1}$. Si l'azimut associé à la colonne du tableau est compris dans $\mathbf{Pl}_{A,1}$, la case du tableau est remplie par le profil distance $x_{A,1}$ auquel on a appliqué le gabarit associé à l'azimut de la colonne. Pour chaque profil distance, c'est-à-dire les lignes du tableau, on répète le même processus. La figure 5.8 résume la construction du tableau correspondant au cas pris en exemple. Bien entendu dans la réalité, ce tableau sera de dimension beaucoup plus grande.

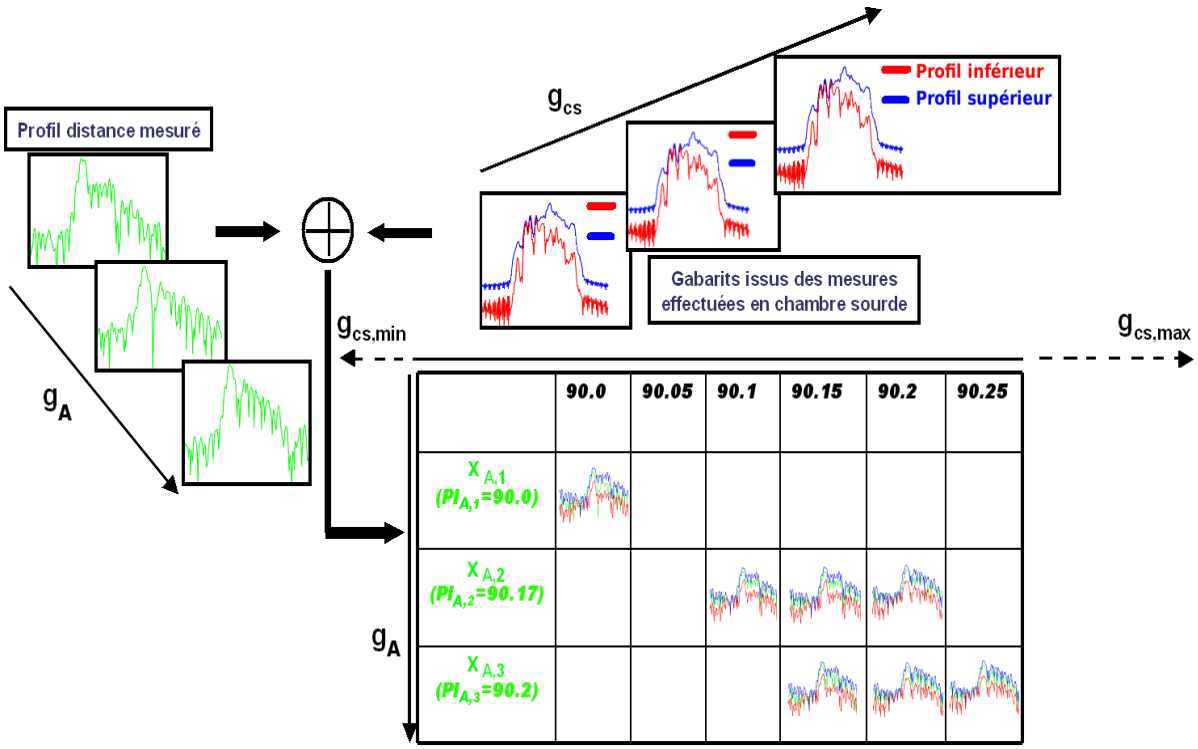


FIG. 5.8 : Construction des données d'apprentissage

5.2.2 Etape 2 : mise en forme du profil distance sous test

Lorsqu'un profil distance est mesuré par le radar, la première étape qui est effectuée dans notre algorithme est le calcul de la densité de possibilité associée. Comme on l'a vu dans la section 1.3.2.3, la densité de possibilité décrit comment la valeur réelle d'un paramètre peut être distribuée, étant donné la mesure de ce paramètre et l'estimation de sa précision. Dans le cas d'un profil distance, le paramètre est la valeur de SER dans une case distance. Dire que l'on calcule la densité de possibilité associée au profil distance de test est en fait un abus de langage car on va plutôt calculer la densité de possibilité associée à la valeur de SER dans chaque case distance du profil distance de test.
Soit $\boldsymbol{x}_{T,l}$ un profil distance de test. On note $\mathbf{xlin}_{T,l}$ ce même profil distance exprimé en linéaire. On fait l'hypothèse qu'en linéaire les échantillons de bruit du profil distance suivent la même distribution. En calculant la fonction de répartition des échantillons de bruit, on peut retenir quatre valeurs (q_1, q_2, q_3, q_4) qui caractérisent la répartition de ces échantillons.

La densité de possibilité $\mathbf{DPlin}_l$ associée au profil distance en linéaire est donc égale à :

$$\mathbf{DPlin}_l = \{\mathbf{DPlin}_l^1, \mathbf{DPlin}_l^2, \mathbf{DPlin}_l^3, \mathbf{DPlin}_l^4\} \tag{5.11}$$

avec :

$$\begin{cases} \mathbf{DPlin}_l^1 = \mathbf{xlin}_{T,l} + q_1 \\ \mathbf{DPlin}_l^2 = \mathbf{xlin}_{T,l} + q_2 \\ \mathbf{DPlin}_l^3 = \mathbf{xlin}_{T,l} + q_3 \\ \mathbf{DPlin}_l^4 = \mathbf{xlin}_{T,l} + q_4 \end{cases} \tag{5.12}$$

Une fois la densité de possibilité calculée pour le profil distance en linéaire, on passe à l'échelle logarithmique pour obtenir la densité de possibilité du profil distance de test en dB que l'on note $\mathbf{DP}_l$.

$$\mathbf{DP}_l = 10 \log 10(\mathbf{DPlin}_l) \tag{5.13}$$

On a donc, pour chaque case distance du profil distance de test, défini une densité de possibilité trapézoïdale (représentée sur la figure 5.9) à partir de ces 4 valeurs. Soit $DP_{l,i}$ la densité de possibilité associée à la i-ème case distance du profil distance de test $\boldsymbol{x}_{T,l}$.

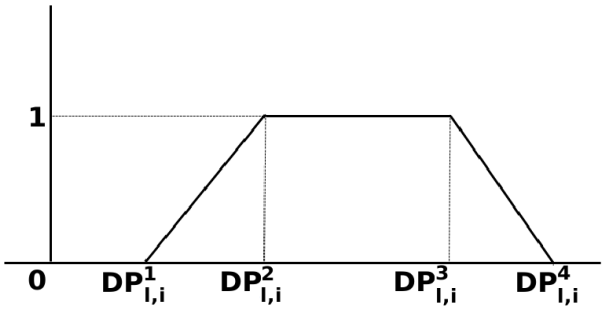


FIG. 5.9 : Densité de possibilité

Ces densités de possibilité associées à chaque case distance du profil distance de test seront comparées aux fonctions d'appartenance associées à chaque case distance des profils distance d'apprentissage pour calculer une possibilité pour chaque case distance. Le calcul de ces fonctions d'appartenance est détaillé dans le paragraphe 5.2.1.

5.2.3 Etape 3 : comparaison du profil distance sous test avec la base d'apprentissage

Cette troisième étape consiste à comparer le profil distance sous test avec les profils distance de la base d'apprentissage. Comme on l'a vu dans les précédentes sections, on compare en fait la densité de possibilité associée au profil distance sous test avec les fonctions d'appartenance construites à partir des profils distance d'apprentissage et des données acquises en chambre sourde.

Sélection des profils distance d'apprentissage

Cependant, on ne va pas comparer la densité de possibilité du profil distance sous test avec toutes les fonctions d'appartenance construites. On va sélectionner dans le tableau des données d'apprentissage (cf. figure 5.8), les fonctions d'appartenance pouvant correspondre au profil distance

sous test en fonction de la valeur de son azimut et de l'écart-type associé. Le profil distance de test $\boldsymbol{x}_{T,l}$ a un angle de gisement noté $g_{T,l}$ et un écart-type associé $\sigma_{g_T,l}$. On définit dans un premier temps une plage de gisement $\mathbf{Pl}_{T,l}$ associée au profil distance sous test :

$$\mathbf{Pl}_{T,l} = \left[g_{T,l} - \sigma_{g_T,l}, g_{T,l} + \sigma_{g_T,l}\right] \tag{5.14}$$

On va ensuite chercher tous les profils distance d'apprentissage qui ont un angle de gisement contenu dans cette plage. Si on reprend le tableau de la figure 5.8, cela consiste à sélectionner les lignes du tableau qui nous intéressent. Prenons comme exemple un profil distance de test avec $g_{T,l} = 90.15°$ et$\sigma_{g_T,l} = 0.1$. Dans ce cas, les deux dernières lignes du tableau 5.8 sont sélectionnées (cf. figure 5.10).

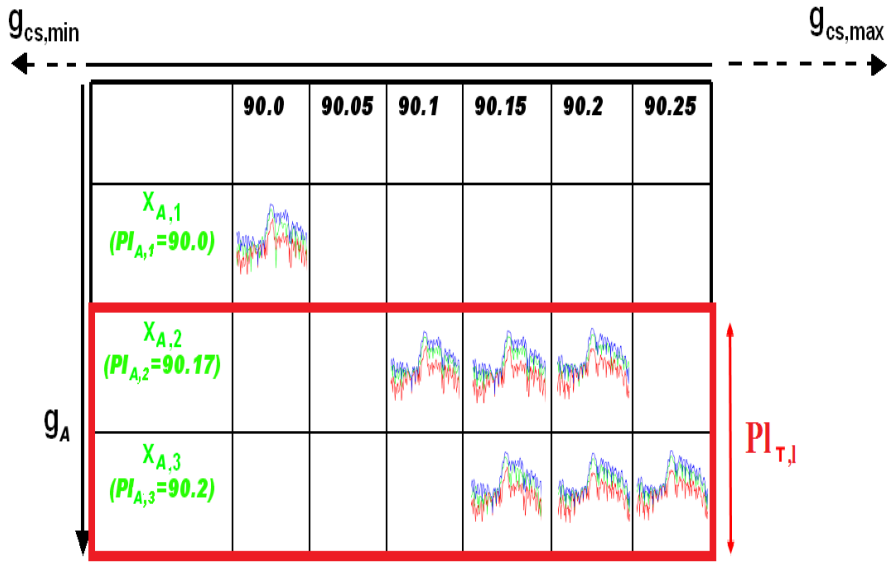


FIG. 5.10 : Sélection des fonctions d'appartenance

Une fois que l'on a sélectionné les lignes, toutes les cases non vides de ces lignes seront traitées de manière à prendre en compte l'incertitude sur l'estimation des azimuts des profils distance d'apprentissage. On compare donc la densité de possibilité associée au profil distance sous test avec toutes les fonctions d'appartenance contenues dans les cases du tableau sélectionnées.

Sélection des cases distance

Lorsqu'on dit que l'on va comparer une densité de possibilité avec des fonctions d'appartenance, il s'agit en fait de calculer case distance par case distance les possibilités à partir de la fonction d'appartenance et de la densité de possibilité dans chaque case distance. Seules les cases distance pour lesquelles la valeur de la SER du profil distance sous test ou la valeur maximum de la fonction d'appartenance dépassant un seuil de détection sont prises en compte. Le seuil de détection est calculé de manière à ne pas

traiter trop d'échantillons de bruit, par exemple une erreur est autorisée par profil distance. Le seuil de détection est calculé de la manière suivante :

$$s_D = \max(Nb_{test}, Nb_{app}) + s_B. \tag{5.15}$$

Les termes Nb_{test} et Nb_{app} sont respectivement le niveau de bruit du profil de test et le niveau de bruit du profil d'apprentissage. Le terme s_B est le seuil au bruit. C'est l'ajustement de ce terme qui permet de limiter le nombre d'échantillons de bruit thermique pris en compte lors de la comparaison des profils distance.

Calcul de la possibilité

Une fois la sélection des cases distance faite, on peut calculer la possibilité pour les cases distance correspondantes. La possibilité dans la case distance i est égale à :

$$P_{l,p,q,i} = \max\{\min[FA_{p,q,i}, DP_{l,i}]\} \tag{5.16}$$

Calcul de la proportion de cases distance avec une possibilité trop faible

On fait l'hypothèse que l'on rejette une case distance lorsque la possibilité dans cette case distance est inférieure à une certaine valeur P_{dec}. La probabilité d'erreur P_{err} peut être exprimée en fonction de ce paramètre.

En fixant, une probabilité d'erreur (par exemple à 5%), on veut trouver les seuils d'acceptation ou de rejet pour les cases distance. Une logique de décision peut être la suivante :

- La classe est rejetée dès que le nombre de cellules M du profil distance avec $P^k_{l,p,q,i} < P_{dec}$ est supérieur à M_{rejet}.
- La classe est acceptée dès que le nombre de cellules M du profil distance avec $P^k_{l,p,q,i} < P_{dec}$ est inférieur à M_{accept}.

avec :

- M_{rejet} tel que $P_{err}(M_{rejet})$ soit inférieur à la probabilité d'erreur fixée.
- M_{accept} tel que $P_{err}(M_{accept})$ soit plus élevée.

On a donc $M_{accept} \leq M_{rejet}$.

Les seuils de décision en pourcentages peuvent donc être choisis comme suit :

- $\alpha_{accept} = \frac{M_{accept}}{N}$
- $\alpha_{reject} = \frac{M_{reject}}{N}$

avec N, le nombre total de cases distance sélectionnées.

Prise de décision

Si le pourcentage de cases distance avec une possibilité inférieure à P_{dec} est supérieur à α_{reject}, on rejette le profil distance. Si le pourcentage de cases distance avec une possibilité inférieure à P_{dec} est inférieur à α_{accept},

on accepte le profil distance. Entre les deux, on ne prend pas de décision. Soit $D_{l,p,q}$, la décision prise pour le profil distance de test l par rapport à la fonction d'appartenance construite à partir du profil distance d'apprentissage p et du gabarit pour l'angle de gisement q.

- Acceptation $D_{l,p,q} = 1$
- Indécision $D_{l,p,q} = 0.5$
- Rejet $D_{l,p,q} = 0$

Cette valeur $D_{l,p,q}$ correspond à la décision prise dans une case du tableau 5.10. Pour prendre une décision finale, il faut fusionner l'ensemble des décisions prises dans chacune des cases distance.

On note D_l^k, la décision finale prise quant à l'appartenance du profil distance sous test à la classe k.

$$D_l^k = \max_{p,q}(D_{l,p,q}) \tag{5.17}$$

La décision finale est donc le maximum de toutes les décisions $D_{l,p,q}$ prises (opérateur fusion "OU" dans la logique floue). La figure 5.11 illustre cette étape de prise de décision finale à partir de l'exemple pris pour construire le tableau 5.10.

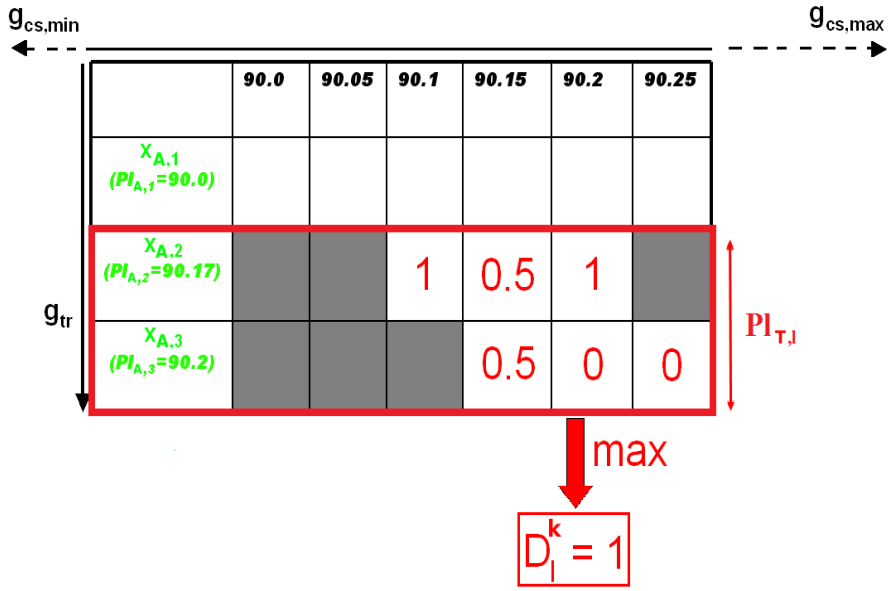


FIG. 5.11 : Prise de décision finale

Bien évidemment, on prend une décision par rapport à une classe. Tout ce qui a été détaillé jusqu'à présent est la décision prise quant à l'appartenance du profil distance sous test à une des classes de la base d'apprentissage. Il faudrait donc répéter l'ensemble du processus pour chaque classe de cible que l'on veut tester. Comme on l'a dit en introduction, on ne possède des gabarits que pour une des classes de cible contenue dans la base de données. Dans un premier temps, on teste donc uniquement l'appartenance à cette classe. Le but de la section suivante est d'expliquer comment on peut adapter les gabarits en notre possession aux autres classes de cible contenues dans la base de données.

5.2.4 Adaptation des gabarits aux autres classes de la base de données

Le but de cette section est d'expliquer la méthode utilisée pour construire des gabarits pour les deux autres classes de cible de notre base de données à partir de ceux que l'on possède pour la classe 1. La première étape est d'adapter les gabarits de la classe 1 afin de pouvoir les utiliser pour construire ceux des autres classes.

Adaptation des gabarits

L'idée est de découper les profils distance en zones caractéristiques et d'appliquer une dynamique constante dans chacune de ces zones. On a choisi de découper un profil distance en zones caractéristiques (par exemple : Pointe de l'avion - Cockpit, Moteurs - Queue de l'avion, etc...).

On fait l'hypothèse que la valeur de la SER dans les cases distance d'une même zone varie avec la même dynamique.

En fonction des dimensions de l'avion, on sélectionne les cases distance correspondantes à chacune des zones caractéristiques. Dans chacune des zones, on calcule la dynamique entre les profils distance supérieurs et inférieur puis on calcule une moyenne (sur les cases distance de la zone) des dynamiques.

Avec les gabarits utilisés jusqu'à présent, dans chaque case distance même à l'intérieur d'une même zone les valeurs de dynamique pouvaient être différentes. Avec le gabarit adapté, dans chaque zone, on applique la moyenne des dynamiques observées. On a donc dans chaque zone des dynamiques constantes. Il paraît évident que l'on risque de dégrader les performances de l'algorithme en procédant de la sorte, cependant cela va nous permettre de construire des gabarits pour les autres classes, certes avec moins de finesse mais cela peut être suffisant pour prendre une décision quant à l'appartenance aux différentes classes. Il est sûr que l'idéal serait de mesurer en chambre sourde les gabarits pour chaque type cible que l'on souhaite reconnaître mais les coûts engagés seraient très importants sans oublier le fait que les cibles à reconnaître sont dans la majorité des cas des cibles ennemies.

Construction des gabarits pour les autres classes de la base de données

Pour construire les gabarits des deux autres classes, on se sert des dynamiques moyennes observées sur les gabarits en notre possession dans les différentes zones. On applique ces dynamiques de la même manière dans les différentes zones des cibles des deux autres classes. La seule chose qui diffère est la dimension de ces zones (nombre de cases distance) qui va varier en fonction de la classe de la cible.

5.2.5 Performances

Performance de l'algorithme sans adaptation des gabarits

Le tableau 5.5 résume les performances de l'algorithme G-FRI en utilisant uniquement des gabarits pour la classe 1 et ne testant donc que

l'appartenance à cette classe. Le taux d'erreur est donné dans le tableau 5.6.

		MA	**MI**	**MR**
		1	1	1
Données réelles	1	89.4	6.4	4.2
	2	56.2	11.4	32.4
	3	5.9	0	94.1

TAB. 5.5 : Matrices de décision **MA**, **MI** et **MR** pour les données réelles et uniquement pour la classe 1 - Algorithme G-FRI

	Taux d'erreur
Classe 1	4.2

TAB. 5.6 : Taux d'erreur - Algorithme G-FRI

On a bien un taux d'erreur proche du taux d'erreur souhaité. L'information donnée par le taux de succès et le taux de bonne identification n'est pas forcément très significative ici car on ne teste l'appartenance qu'à une seule classe.

Performance de l'algorithme avec adaptation des gabarits

Cette version de l'algorithme est celle utilisant des gabarits pour les trois types de classes. Les gabarits pour les classes 1, 2 et 3 ont été adaptés à partir des gabarits initiaux de la classe 1.

		MA			**MI**			**MR**		
		1	2	3	1	2	3	1	2	3
Données réelles	1	95.7	80.8	2.1	0	8.5	0	4.3	10.6	97.9
	2	67.1	96.2	20.2	10.1	1.3	10.1	22.8	2.5	69.6
	3	10.5	20.7	92.0	0	3.4	2.0	89.5	75.9	6.0

TAB. 5.7 : Matrices de décision **MA**, **MI** et **MR** pour les données réelles - Algorithme G-FRI

On peut calculer les taux d'erreur, taux de succès et taux de bonne identification pour l'algorithme G-FRI avec adaptation des gabarits. Les résultats sont donnés dans le tableau 5.8. On peut dans un premier temps comparer les résultats pour la classe 1 par rapport à ceux obtenus avec l'algorithme G-FRI sans adaptation des gabarits. On se rend compte que l'adaptation des gabarits dégrade les performances par rapport à celles obtenues sans adaptation. En effet, on a plus de profils distance des classes 2 et 3 (respectivement 10 points et +5 points) qui sont déclarés comme

appartenant à la classe 1. De manière symétrique (le nombre de cibles incertaines restant à peu près similaire entre les deux versions), le nombre de profils distance des classes 2 et 3 pour lesquels on a rejeté la classe 1 diminue entre les 2 versions. C'est le type de résultats auxquels on s'attendait. Il existe une dégradation mais cette dégradation reste raisonnable. On peut donc considérer qu'en l'absence de données en chambre sourde pour certaines cibles, la méthode d'adaptation des gabarits est applicable. Dans un cas opérationnel, on aurait bien sûr utilisé les gabarits enregistrés en chambre sourde sans adaptation pour les cibles de la classe 1 et utilisé l'adaptation des gabarits pour les cibles des classes 2 et 3.

	TE	TBI	TS
Données réelles	4.0	30.1	37.5

Tab. 5.8 : Taux d'erreur, taux de succès et taux de bonne identification - Algorithme G-FRI

En terme de taux d'erreur, cette deuxième version de l'algorithme ne dégrade pas les performances. Dans la première version de l'algorithme, les taux de succès et d'identification n'étaient pas révélateurs des performances de l'algorithme car l'appartenance à une seule classe était testée, donc il n'y avait pas de possibilité d'avoir des décisions multiples. Dans cette deuxième version, l'appartenance aux 3 classes est testée. L'information donnée par les taux de succès et taux de bonne identification a donc plus d'importance. Tout d'abord, on se rend compte que les taux d'erreur pour les 3 classes restent proches de la valeur fixée des 5%. Le taux d'erreur globale est de 4%, on a donc un contrôle du taux d'erreur avec ce jeu de données. Les taux de succès restent quant à eux relativement bas pour les classes 1 et 2 (respectivement 22.8% et 14.9%). Le taux de bonne identification est encore plus faible avec respectivement 11.4% et 6.4% pour les classes 1 et 2. Pour la classe 3, le taux de succès est de 82%, ce qui est un taux plutôt satisfaisant, d'autant plus que le taux de bonne identification est égal au taux de succès. Les performances sont tout de même nettement meilleures que celles obtenues avec l'algorithme D-FRI car tout en gardant le contrôle du taux d'erreur, les taux de succès global et taux de bonne identification global ont été augmentés respectivement de 37.5 points et 30.1 points avec ce nouvel algorithme.

5.3 Conclusion du chapitre

Le but de ce chapitre était de présenter des algorithmes utilisant les principes de la logique floue et permettant de se rapprocher au maximum des objectifs de la thèse (maîtrise du taux d'erreur et maximisation du taux de succès). Le premier algorithme présenté dans la section 5.1 permet de contrôler le taux d'erreur avec les données synthétiques et réelles néanmoins le taux de succès obtenu avec cet algorithme n'est pas satisfaisant notamment avec les données réelles.

Un deuxième algorithme a donc été présenté dans la section 5.2 pour améliorer les performances en terme de taux de succès tout en gardant

la maîtrise sur le taux d'erreur. Cet algorithme prend en compte l'angle de gisement et sa précision lors de la construction de la fonction d'appartenance, ce qui est une des explications possibles des mauvaises performances obtenues avec l'algorithme D-FRI sur les données réelles. On utilise également des données supplémentaires issues de mesure en chambre sourde pour affiner la construction des fonctions d'appartenance. L'algorithme G-FRI est au final beaucoup plus complexe que l'algorithme D-FRI mais il permet d'améliorer sensiblement les performances en terme de taux de succès tout en gardant une maîtrise du taux d'erreur.

Nous avions en notre possession des mesures en chambre sourde pour une seule des cibles présentes dans la base d'apprentissage. Nous avons donc également proposé une méthode permettant d'adapter l'information fournie par ces mesures en chambre sourde aux autres cibles de la base d'apprentissage. Même si cette méthode d'adaptation dégrade légèrement les performances, elle permet néanmoins d'utiliser cette deuxième version de l'algorithme en utilisant un nombre limité de mesures en chambre sourde. Bien évidemment, plus le nombre de classes de cibles pour lesquelles nous disposons de mesures en chambre sourde est important, meilleures seront les performances de l'algorithme G-FRI.

Au final, il est intéressant de prendre en compte explicitement ce degré de fiabilité des profils distance formant la base d'apprentissage. Cet algorithme permet une amélioration progressive des performances de classification par l'introduction de profils distance de plus en plus fiables dans la base d'apprentissage, sans avoir à modifier la nature des traitements.

Parallélisation sur GPU

6

Sommaire

6.1 Les processeurs graphiques (GPU) 106
6.1.1 Historique : vers une architecture many-cores 106
6.1.2 Puissance de calcul des GPUs 107
6.1.3 Modèle de programmation sur GPUs 110
6.1.4 Accès mémoire 111
6.1.5 Génération Tesla vs génération Fermi 112
6.2 Etat de l'art de l'utilisation des GPUs dans les domaines en lien avec la thèse 113
6.2.1 Calcul de la SER sur GPU 113
6.2.2 Reconnaissance de cibles sur GPU 113
6.2.3 K plus proches voisins sur GPU 114
6.2.4 Méthodes probabilistes sur GPU 115
6.2.5 Logique floue sur GPU 116
6.3 Algorithme des KPPV et parallélisation sur GPU 117
6.3.1 Nombre d'opérations à virgule flottante 118
6.3.2 Temps de calcul sous Matlab 118
6.3.3 Découpage en thread sur l'architecture many-core GPU . 119
6.3.4 Latence, Occupation et parallélisme 119
6.3.5 Implémentation CUDA C++ sur GPU 123
6.3.6 Implémentation OpenCL 129
6.3.7 Implémentation Matlab mex-CUDA et Matlab mex-OpenCL 129
6.3.8 Conclusion 130
6.4 Méthodes probabilistes et parallélisation sur GPU . . . 131
6.5 Algorithme logique floue et parallélisation sur GPU . 134
6.5.1 Algorithme D-FRI 134
6.5.2 Algorithme G-FRI 135
6.5.3 Accélération de l'étape de calcul des possibilités 137
6.6 Conclusion du chapitre 138

Le but de ce chapitre est de présenter les aspects parallélisation des différents algorithmes vus jusqu'à présent. Après une brève introduction sur les GPUs dans la section 6.1, nous proposons dans la section 6.2 un état de l'art des solutions GPU existantes dans les domaines étudiés durant la

thèse. Ensuite, la section 6.3 présente une implémentation sur GPU de l'algorithme des KPPV présenté dans le chapitre 3 et une étude détaillée des concepts nécessaires à une parallélisation efficace sur ce type de technologie. Dans la section 6.4, nous étudions le coût de calcul des algorithmes de type probabiliste présentés dans le chapitre 4 et nous réfléchissons brièvement à la manière de les paralléliser. Enfin, dans la section 6.5, nous nous interrogeons sur le coût de calcul engendré par les algorithmes de reconnaissance de cibles basés sur la logique floue présentés dans le chapitre 5 et sur la faisabilité d'une mise en oeuvre opérationnelle sur GPU de ces algorithmes. Nous détaillons également les outils utilisés durant la thèse pour accélérer ces deux algorithmes.

6.1 Les processeurs graphiques (GPU)

Les GPUs (Graphics Processing Unit en anglais) ou processeurs graphiques sont des microprocesseurs présents sur les cartes graphiques des PCs, dans les consoles de jeux ou depuis quelques années sur les serveurs de calcul. Les processeurs des cartes graphiques modernes ont une structure hautement parallèle qui les rend efficaces pour une large palette de tâches graphiques mais également pour les tâches de calcul. C'est ce dernier point qui nous intéresse dans le cadre de la thèse. Les trois grands fabriquants de GPU sont NVIDIA, ATI/AMD et Intel. Les deux premiers fabriquants sont issus du monde du rendu graphique et proposent depuis un certain nombre d'années maintenant des gammes de cartes dédiées spécifiquement aux calculs scientifiques. Intel, quant à lui, est un fabricant historique de solutions multi-core de type CPU mais il propose également des processeurs graphiques de faible puissance et à faible coût qui sont généralement intégrés sur les cartes mères de nos ordinateurs. Cependant, depuis plusieurs années et le projet Larrabee, Intel s'oriente de plus en plus vers des solutions many-cores qui ont pour but de concurrencer NVIDIA et ATI/AMD dans le secteur du calcul haute performance. Dans le cadre de notre thèse, nous avons utilisé uniquement des GPUs de marque NVIDIA. Les concepts mis en avant sont transposables sur les GPUs des autres fabricants.

6.1.1 Historique : vers une architecture many-cores

A l'origine, les puces graphiques étaient uniquement composées de pipelines graphiques à fonction fixe notamment pour le rendu de volume avec la projection de polygones d'un espace 3D sur un écran 2D. Dans les années 1990, apparaîssent des cartes graphiques dédiées à la gestion et l'affichage d'éléments représentés en 3 dimensions, notamment avec le fabricant 3DFX (plus tard racheté par NVIDIA). Jusqu'en 1996 et la sortie des première cartes intégrées 2D/3D par ATI, les cartes 2D étaient proposées séparément des cartes dites accélératrice 3D, chacune ayant un processeur graphique spécifique. A partir du milieu des années 1990 et l'arrivée de NVIDIA sur le marché, les cartes graphiques ont énormément évolué devenant de plus en plus complexes et spécialisées mais leurs puissances de calcul restant exploitées uniquement pour les jeux vidéo.

Au début des années 2000, certains chercheurs, notamment dans les domaines des simulations numériques (Krüger and Westermann 2003, Bolz et al. 2003) ou de l'imagerie médicale (Lefohn et al. 2003), commencent à s'intéresser à la puissance de calcul des GPUs qui pour un coût très réduit permettent d'exécuter des opérations de calcul général. Les GPUs leur permettent d'obtenir des performances bien meilleures que celles obtenues avec la simple utilisation du CPU. La recherche scientifique profite de cette nouvelle technologie explorant des domaines jusqu'à présent limités par les capacités de calculs. C'est à cette époque que naît le mouvement appelé GPGPU (General Purpose computing on GPUs). Le problème de ces premiers GPUs est qu'ils exigent l'utilisation des langages de programmation graphique tels que OpenGl ou Cg afin d'assurer la comptabilité avec le CPU. Les utilisateurs devaient donc programmer leurs applications scientifiques en utilisant les langages de programmation graphique, ce qui a limité pendant un certain temps l'accessibilité aux GPUs.

C'est dans le but de rendre plus accessible la programmation sur GPU que NVIDIA a créé en 2006 l'architecture CUDA (Compute Unified Device Architecture). La même année ATI crée la technologie ATI Stream qui permet également d'exécuter des calculs génériques sur processeurs graphiques mais ce n'est qu'en 2008 qu'elle devient pleinement utilisable. Les GPUs ont donc été modifiés (en transformant certains étages du pipeline graphique en coeurs de calcul) afin de faciliter leurs programmations pour les applications scientifiques et pour les rendre compatibles avec des langages à hautes performances comme C et C++. L'appellation CUDA (ou ATI Stream) regroupe donc l'architecture matérielle et le modèle de programmation parallèle. Les programmeurs peuvent choisir d'exprimer le parallélisme avec des langages à hautes performances comme C, C++ et Fortran en utilisant des API comme CUDA (NVIDIA), ATI Stream (ATI/AMD) ou OpenCL (standard non propriétaire).

6.1.2 Puissance de calcul des GPUs

Pour suivre la demande grandissante d'applications aux coûts de calcul de plus en plus importants, les GPUs ont évolué vers des processeurs hautement parallèles, multi-coeurs et multi-threads ayant une puissance de calcul et une bande passante mémoire très élevée (cf. figures 6.1 et 6.2).

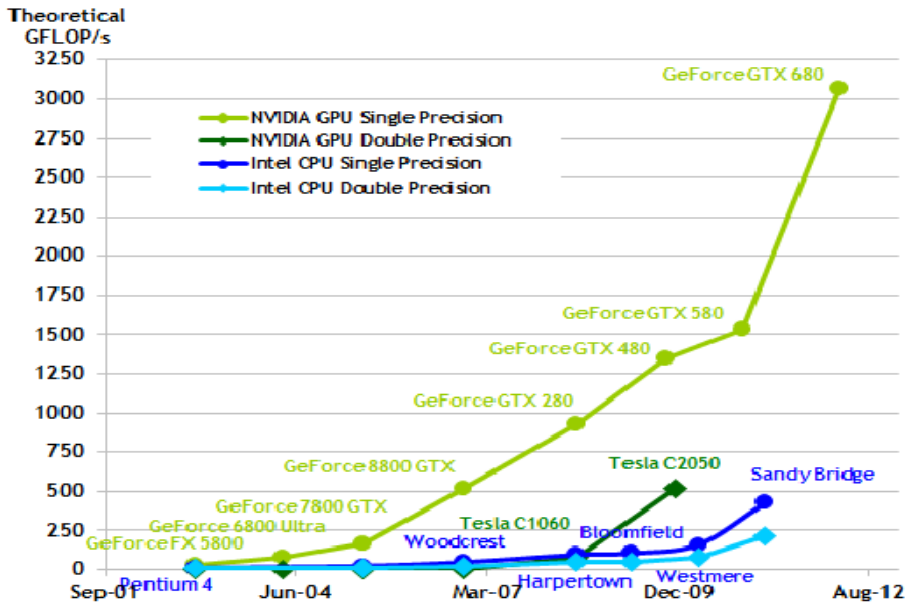


FIG. 6.1 : Puissance de calcul GPU (NVI 2012)

La puissance de calcul est évaluée en terme de nombre d'opérations à virgule flottante (ou FLOP) par seconde, on parle alors de FLOPS pour FLOP/s. Les opérations à virgule flottante incluent toutes les opérations qui impliquent des nombres réels. Il existe deux principaux niveaux de précision pour les nombres réels :

- simple précision (SP) sur 32 bits ;
- double précision (DP) sur 64 bits.

Dans le cadre de notre thèse, nous utilisons des nombres réels en simple précision, ce qui est largement suffisant pour les traitements mis en oeuvre.

Les cartes que nous avons utilisées durant notre thèse sont des cartes de la génération Fermi (C2050) et de la génération Tesla (C1060) qui présentent des performances pics en simple précision respectivement de 1.03$TFlops$ et de 622$GFlops$). En 2013, la nouvelle génération Kepler présente des performances plus élevées (près de 4$TFlops$ en simple précision et 2700 coeurs pour la K20X) mais avec une architecture restant dans les grandes lignes proche de celle de Fermi.

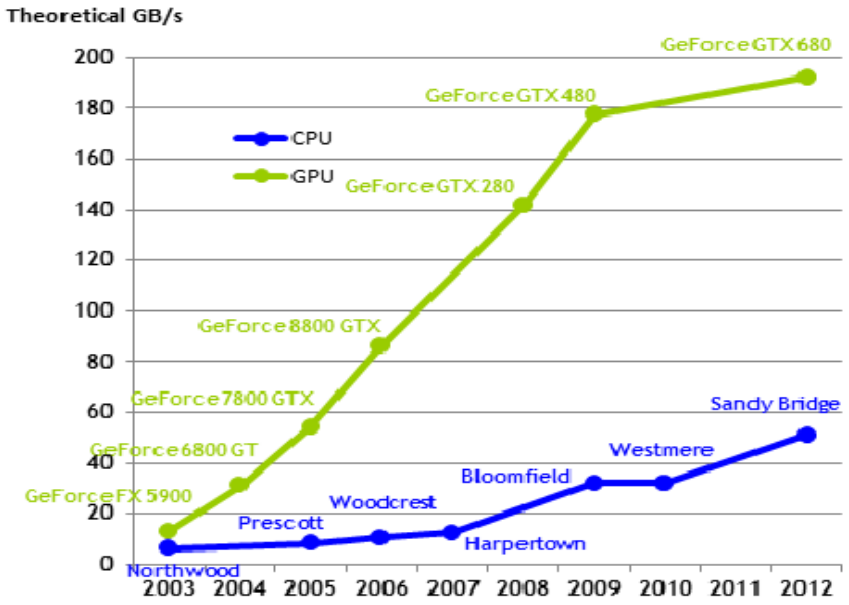


FIG. 6.2 : Débit mémoire sur GPU (NVI 2012)

La différence de performance entre GPU et CPU est due à leur architecture. Les CPUs et les GPUs ont une utilisation généralement très différente. On attend d'un CPU qu'il exécute une tâche le plus rapidement possible (calcul séquentiel) alors que l'on attend d'un GPU de pouvoir traiter une tâche sur un maximum de données dans un temps réduit (calcul parallèle). Cela induit, dans le cas d'un GPU, une multiplication des unités de traitement et dans le cas du CPU une complexification des unités de contrôle ainsi qu'une augmentation régulière de la mémoire cache embarquée (cf. figure 6.3).

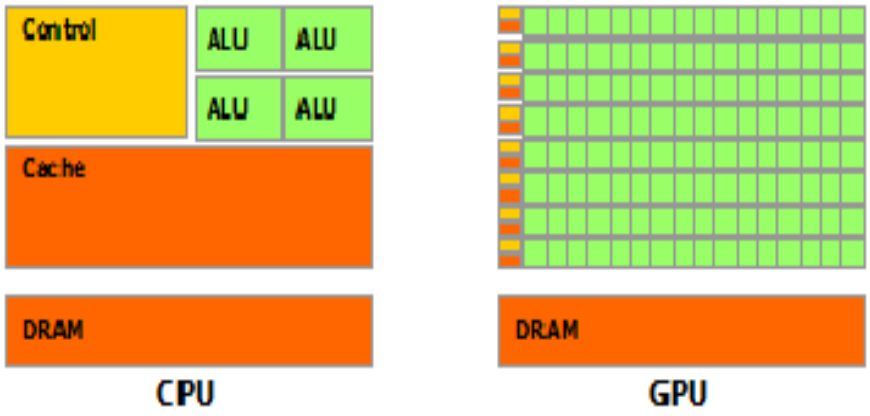


FIG. 6.3 : Architecture CPU et GPU (NVI 2012)

6.1.3 Modèle de programmation sur GPUs

Il existe trois principaux modèles de programmation sur GPUs :
- le modèle CUDA (Common Unified Device Architecture) ;
- le modèle ATI Stream ;
- l'architecture OpenCL (Open Computing Language).

CUDA

CUDA est un modèle de programmation SIMT (Single Instruction Multiple Threads) des architectures "many cores" du fabriquant NVIDIA (NVI 2012). CUDA permet de programmer en C, C++ ou Fortran. Le principe est de définir un "kernel" comme étant le programme à exécuter en parallèle sur les coeurs du GPU. Chacune de ces instances s'appelle un "thread". Un même "kernel" est exécuté en parallèle par tous les threads. Avec l'architecture CUDA, les threads sont organisés de manière hiérarchique afin de correspondre à l'architecture par blocs de coeurs de calcul, appelée "Streaming Multiprocessor" (SM), des GPUs (32 coeurs par blocs pour l'architecture Fermi). Ils sont en effet regroupés en "blocs de threads", eux mêmes regroupés en "grilles de blocs de threads". Avec l'architecture Fermi, le nombre de threads par bloc peut aller jusqu'à 1024. N'importe quel programme CUDA peut être divisé en deux parties : le code exécuté sur le GPU et le code exécuté sur le CPU. Ce dernier est en grande partie dédié aux transferts de mémoire entre GPU et CPU.

ATI Stream

La Technologie ATI Stream, précédemment appelée Close to Metal, basée sur Brook+ (qui est une extension du langage ANSI C, développée par l'université Stanford), est une technologie informatique développée par AMD permettant de programmer les processeurs graphiques ATI. Nous ne rentrerons pas dans les détails de cette technologie car nous n'avons pas utilisé de cartes graphiques ATI durant la thèse mais on peut tout de même dire qu'elle reprend les mêmes concepts que CUDA.

OpenCL

OpenCL est proposé comme un standard ouvert par le consortium Khronos Group, créé en 2006 par des sociétés telles que Intel, Apple, ARM, ATI ou NVIDIA pour les plus connues. Leur objectif est de créer un couple langage/API commun pour les architectures parallèles dont les spécifications sont rendues publiques et sont utilisables gratuitement. OpenCL propose donc un modèle de programmation se situant à l'intersection entre le monde des CPUs et des GPUs, les premiers étant de plus en plus parallèles, les seconds étant de plus en plus programmables. OpenCL reprend les grands principes du modèle de programmation CUDA (kernel, thread, bloc de threads et grille de blocs appelés respectivement kernel, work-item, work-group et NDrange en OpenCL). OpenCL a néanmoins des objectifs plus larges car le modèle n'est pas uniquement dédié aux GPUs. OpenCL permet notamment de tirer partie

de la puissance des GPUs, des CPUs multi-coeurs ou d'autres systèmes de calcul intensifs tels que le CELL d'IBM ou encore plus récemment des processeurs intégrés many-core Xeon Phi d'Intel et tout cela via une infrastructure de programmation unique.

Durant notre thèse, nous avons principalement utilisé l'architecture CUDA pour programmer sur les cartes graphiques NVIDIA dont nous disposions. Nous nous sommes tout de même un peu intéressés au standard OpenCL, dans un premier temps pour comparer les performances de ce standard sur les cartes NVIDIA par rapport à celles obtenues en utilisant CUDA (cf. section 6.3.6) puis également pour paralléliser (sur les coeurs CPU ou GPU) l'étape de calcul des possibilités pour l'algorithme G-FRI (cf. section 6.5.3).

6.1.4 Accès mémoire

Les performances d'un programme GPU sont très influencées par la gestion mémoire (Kirk and Hwu 2010). La mémoire est organisée hiérarchiquement et il existe plusieurs solutions pour accéder aux données (cf. figure 6.4).

Chaque thread a sa propre mémoire, les registres. Chaque bloc a une mémoire partagée entre tous les threads, la mémoire "shared". Chaque thread de n'importe quel bloc peut également accéder à la mémoire globale du GPU.

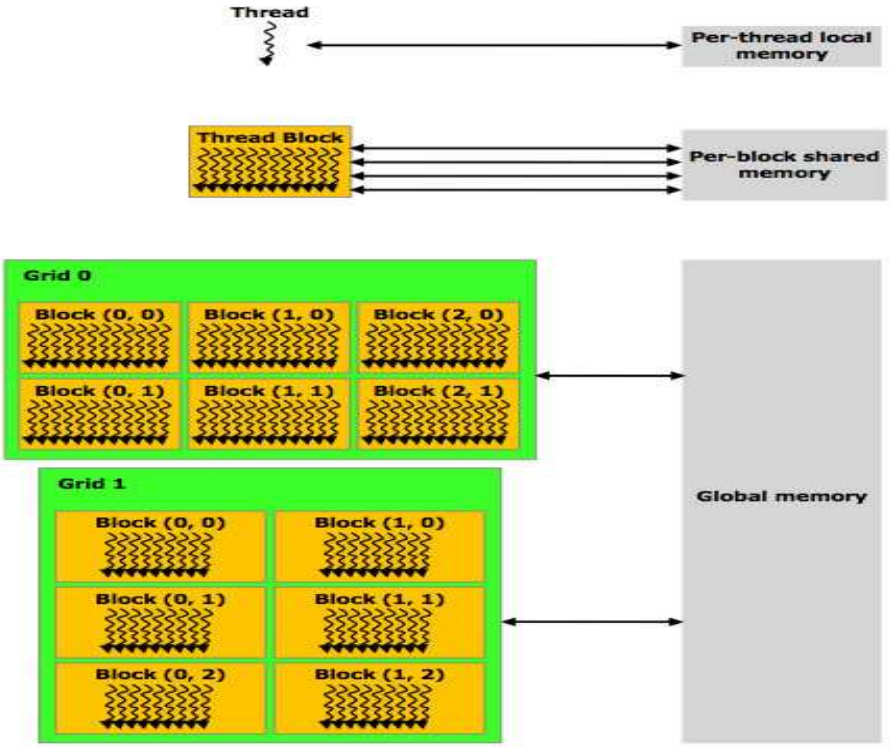


FIG. 6.4 : Architecture mémoire des GPUs (NVI 2012)

Pour accéder à cette mémoire globale plus rapidement, il existe deux

"caches" : le "cache texture" et le "cache constante", et sur l'architecture Fermi des caches L1 et L2 standard (cf. 6.5).

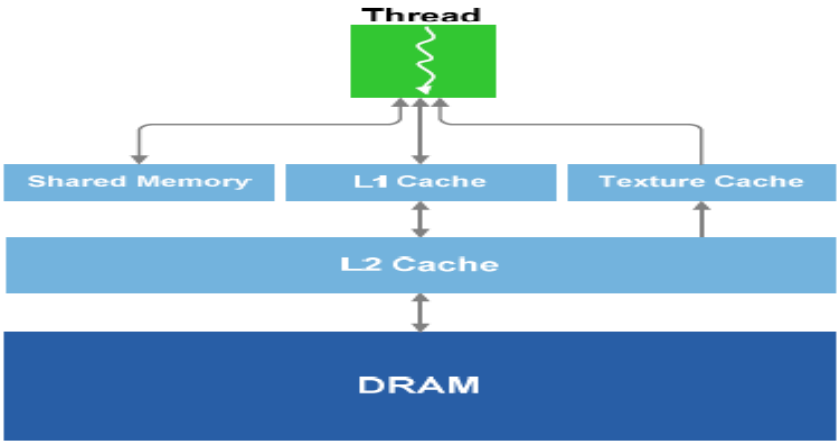


FIG. 6.5 : Architecture mémoire des GPUs Fermi (NVI 2012)

Il convient également de faire remarquer qu'avec la base de données utilisée dans le cadre de la thèse, les problèmes de transferts de mémoire entre la carte graphique et le PC qui peuvent potentiellement créer un goulot d'étranglement ne nous concernent pas. En effet, la capacité mémoire des cartes graphiques (1 à 4 Go) permet de stocker entièrement la base d'apprentissage sur la carte graphique et peut être chargée offline.

6.1.5 Génération Tesla vs génération Fermi

Les cartes GPU utilisées dans notre thèse sont des cartes NVIDIA de deux générations différentes :
- la NVIDIA C2050 de génération Fermi (1.03 TFlops) ;
- la NVIDIA C1060 de génération Tesla (622 GFlops).

La principale innovation des GPUs de la génération Fermi est la présence des caches L1 et L2 et une augmentation de la taille de la mémoire "shared". Avec la génération Fermi, chaque Streaming Multiprocessor (constitué de 32 coeurs de calcul) dispose de 64*Ko* de RAM configurable entre la mémoire "shared" et le cache L1. Lorsque par exemple, on utilise 48*Ko* de mémoire "shared", il reste 16*Ko* de mémoire pour le cache L1 et *vice versa*. Une autre amélioration est l'ordonnancement des warps [1] exécutés par les Streaming Multiprocessor (SM). Avec la génération Fermi, chaque SM possède deux ordonnanceurs de warps et deux unités de distribution d'instructions, ce qui permet de traiter et d'exécuter deux warps en parallèle, ce qui n'était pas le cas avec les générations antérieures.

[1] Un warp est un groupe de 32 threads parallèles.

6.2 Etat de l'art de l'utilisation des GPUs dans les domaines en lien avec la thèse

Il existe aujourd'hui de nombreux domaines d'application utilisant la puissance de calcul des GPUs. En effet, depuis plusieurs années les GPUs sont devenus de plus en plus puissants et programmables (Owens and al 2007). Ce développement a intensifié l'utilisation des GPUs et permis des accélérations d'un à deux ordres de grandeur dans de nombreux domaines d'applications comme les simulations physiques (Cooke et al. 2011), la compression d'image (Lin and Lin 2010), le traitement d'image (Zhang et al. 2010) les réseaux de neurones (Yudanov et al. 2010, Nabiyouni and Aghamirzaie 2012), les simulations Monte-Carlo (Yang et al. 2012, Chitchian et al. 2013) ou encore la tomographie (Xu and Mueller 2007, Liria et al. 2012).

6.2.1 Calcul de la SER sur GPU

Dans le domaine radar, on trouve plusieurs travaux sur la prédiction de la SER accélérée sur GPU. En 2010, Tao et al. (Tao et al. 2010) proposent d'implémenter complétement sur GPU (avec l'architecture CUDA) un algorithme de lancer de rayon pour prédire la valeur de la SER de cibles complexes telles que des missiles, des bateaux et des avions de chasse. Même si l'étape de calcul de la SER n'est pas un sujet d'étude des travaux de thèse présentés dans ce document, il faut savoir que cette étape est indispensable pour obtenir les profils distance synthétiques qui garniront la base d'apprentissage et qu'elle est très coûteuse en temps de calcul. Les résultats montrent que l'algorithme de lancer de rayon permet d'obtenir des prédictions de SER très proches des valeurs réelles et la parallélisation proposée permet d'obtenir des temps de calcul environ 30 fois moins long qu'avec une implémentation sur CPU. Dans le même style, en 2013, Zoric et al. (Zoric et al. 2013) propose une implémentation accélérée de calcul de la SER. Dans cet article, les auteurs utilisent la méthode des moments avec un grand nombre d'excitations (plus de 30000) pour calculer les valeurs de la SER de cibles complexes de type avions de chasse ou hélicoptères. Les auteurs utilisent la librairie d'algèbre linéaire de NVIDIA sur GPU (CUBLAS) pour accélérer les calculs.

6.2.2 Reconnaissance de cibles sur GPU

Plus proche des travaux de thèse, en 2010, Dessauer et al. (Dessauer et al. 2010) proposent une parallélisation sur GPU d'une application de reconnaissance de cible automatique (ou ATR pour Automatic Target Recognition) qui permet d'accélérer les goulots d'étranglement de calcul dans ce type d'application. Une version parallélisée d'une méthode de poursuite de cibles y est présentée avec des gains intéressants. Les applications ATR sont différentes des applications NCTR car elles utilisent principalement des radars de type air-sol et des images SAR pour faire la reconnaissance des cibles.

Dans le domaine NCTR, on ne trouve à notre connaissance actuelle-

ment aucun article utilisant les capacités des cartes GPU pour accélérer les traitements.

6.2.3 K plus proches voisins sur GPU

Dans la littérature, on trouve de nombreux articles traitant de la parallélisation de l'algorithme des K plus proches voisins sur GPU.

En 2009, Kuang et Zhao (Kuang and Zhao 2009) proposent une implémentation sur GPU de l'algorithme des KPPV appliqué à la reconnaissance de formes. Un thread est affecté au calcul de la distance entre un profil de test et un profil d'apprentissage. Le nombre de threads par bloc est fixé à 256. Tous les vecteurs d'attributs nécessaires aux calculs de toutes les distances d'un même bloc sont chargés en mémoire shared afin d'accéder plus rapidement à ces données en mémoire. Autant dire que les attributs doivent être de taille suffisamment petite pour que tous les vecteurs d'attributs nécessaires puissent être stockés en mémoire shared. Les auteurs utilisent également l'algorithme de tri Radix du CUDA sdk pour trier les distances. Par contre, les auteurs préconisent de ne pas utiliser le GPU pour l'étape de décision qui consiste à calculer le nombre d'occurences des différentes classes parmi les K plus petites distances car le CPU est mieux adapté pour accomplir cette étape (le coût de calcul de cette étape est vraiment très faible et elle nécessite un nombre important de branchements).

En 2010, Liang et al. (Liang et al. 2010) proposent d'évaluer la parallélisation de ce type d'algorithme en utilisant l'architecture CUDA et pour une application de classification de texte. Les auteurs proposent de paralléliser le calcul des distances entre les objets de test et les objets de référence (chaque thread calcule une de ces distances). Les chargements mémoire sont le plus souvent possible réalisés pendant que d'autres threads calculent des distances en utilisant les CUDA Stream de manière à réduire le temps d'exécution. L'étape de sélection des K plus proches voisins est également réalisée sur GPU. Toutes les distances calculées dans un bloc de threads sont stockées en mémoire shared et les K plus petites distances sont sélectionnées. Un deuxième kernel est défini pour sélectionner les K plus petites distances sur l'ensemble des blocs. L'étape de décision qui consiste à compter le nombre d'occurences de chaque classe parmi les K plus proches voisins n'est pas effectuée sur GPU. Les auteurs présentent des temps d'exécution environ 45 fois plus rapides pour la version parallèle des KPPV comparée à une version séquentielle.

Des algorithmes de recherche des K plus proches voisins (cf. chapitre 3) ont également été implémentés sur GPU (en CUDA) pour des applications dans le traitement d'image et notamment pour une application de "matching" de caractéristiques de grande dimension (Garcia and al. 2010) ou encore dans le domaine de l'informatique décisionnelle (Huang et al. 2012) avec des performances intéressantes (avec un gain d'environ 65 dans les deux cas). En 2011, Barrientos et al. (Barrientos et al. 2011) proposent également une implémentation sur GPU d'un algorithme de recherche de type KPPV pour trois types d'applications différentes (recherche de texte,

recherche d'images et recherche de visage). Les auteurs présentent des gains de l'ordre de 20 par rapport à des implémentations sur CPU.

6.2.4 Méthodes probabilistes sur GPU

On trouve également plusieurs implémentations parallèles d'algorithmes utilisant les méthodes probabilistes (cf. chapitre 4) et plus précisement pour des algorithmes utilisant des modèles de mélanges de Gaussiennes (GMM). La plupart des articles traitent de l'estimation des paramètres et de l'utilisation de l'algorithme EM. Bien que nous n'ayons pas utilisé l'algorithme EM durant notre thèse (nous étions dans un cas supervisé), il est tout de même intéressant d'étudier brièvement ce qui se fait sur le sujet.

En 2011, Machlica et al. (Machlica et al. 2011) proposent une implémentation sur GPU (en CUDA) de l'estimation des statistiques du GMM via un algorithme EM. L'algorithme est utilisé pour une application de reconnaissance de parole. Les GMM présentés dans cet article sont des GMM avec des matrices de covariances diagonales. Le modèle ne prend donc en compte que la variance des différentes variables constituant les échantillons et pas les covariances entre ces différentes variables. Il s'agit donc d'un modèle relativement simple. Les auteurs définissent quatre kernels correspondant à quatre tâches différentes de l'algorithme EM. Certaines de ces tâches sont dépendantes entre elles, les quatres kernels sont donc exécutés séquentiellement (pas de possibilité de les exécuter en concurrence). La parallélisation se fait sur les composantes du GMM. Le nombre de composantes utilisées dans cet article varie de 32 à 2048 composantes. Lorsque le problème est réduit à seulement quelques composantes, la parallélisation n'est donc pas efficace. On voit déjà bien avec les résultats présentés dans l'article que les performances sont meilleures avec 2048 composantes qu'avec 32 (25% de la puissance pic avec 2048 composantes et 17% avec 32 composantes). Pour le cas étudié dans la thèse (cf. chapitre 4), où seulement trois classes sont modélisées, il serait donc difficile d'utiliser cette parallélisation car le nombre de composantes n'est pas suffisant. Mais pour une application plus réaliste avec une centaine de classes de cibles différentes, cette structure de parallélisation est envisageable. Du point de vue de la gestion mémoire, les auteurs utilisent le cache texture et stockent l'ensemble des résultats intermédiaires en mémoire shared pour améliorer les performances. Les résultats montrent que l'estimation des paramètres sur GPU est 400 fois plus rapide qu'une version standard sur CPU et 130 fois plus rapide qu'une version sur CPU utilisant des instructions SSE.

En 2012, Fuqiu et al. (Fuqiu et al. 2012) présentent une parallélisation sur GPU du calcul de "Super Vecteurs" GMM pour une application de reconnaissance de parole. Ce type d'approche consiste à combiner SVM et GMM en transformant une séquence de vecteurs de longueur arbitraire en un seul vecteur de très grande dimension appelé Super Vecteur. Le Super Vecteur est ensuite utilisé en entrée d'un SVM. Le GMM est utilisé pour modéliser les vecteurs d'attributs puis à partir de ce GMM sont calculées, par exemple, les statistiques d'ordre zéro, un et deux qui sont

ensuite réarrangées pour former le Super Vecteur. La parallélisation se fait également sur le nombre de composantes du GMM qui est fixé à 1024 dans cet article. Le gain de la parallélisation sur GPU présenté dans l'article est de 64 par rapport à une implémentation sur CPU utilisant des instructions SSE.

On trouve de nombreux autres articles sur le sujet. Parmi eux, on peut citer l'article de Gupta et Owens (Gupta and Owens 2009) qui présentent en 2009 un calcul optimisé avec une utilisation intensive de la bande passante mémoire des paramètres d'un GMM. Les auteurs font une étude poussée de trois techniques de réduction de calcul pour la mise en oeuvre d'un GMM et proposent des modifications de celles-ci pour pouvoir utiliser au mieux les caractéristiques des GPU. Enfin, en 2012, Chen et al. (Chen et al. 2012) proposent une parallélisation sur GPU d'un algorithme apprentissage incrémental pour un GMM. Comme dans l'article de Gupta, les auteurs proposent de modifier l'algorithme pour qu'il puisse s'adapter aux caractéristiques des GPU.

6.2.5 Logique floue sur GPU

Dans la littérature, on ne trouve pas d'articles traitant de la parallélisation sur GPU d'algorithmes de reconnaissance de cibles et utilisant les concepts de la logique floue. Même dans d'autres domaines que la reconnaissance de cibles, il existe très peu d'articles abordant la parallélisation sur GPU d'algorithmes basés sur la logique floue. Les algorithmes utilisant les concepts de fonctions d'appartenance, densités de possibilité ou possibilités (cf. chapitre 5) ne sont pas très courant dans la littérature, il est donc encore plus difficile de trouver des articles qui abordent la parallésation de ces algorithmes.

Néanmoins, en lien avec les concepts de logique floue, Srikanthan et al. (Srikanthan et al. 2011) proposent en 2011 une implémentation de l'algorithme Fuzzy C-means sur GPU (en CUDA). Cet algorithme est un algorithme de clustering. La particularité des algorithmes de clustering flous est que les données peuvent être affectées à plus d'une classe et que pour chaque classe, un degré d'appartenance est défini. La parallélisation est réalisée sur l'étape de calcul des fonctions d'appartenance qui sont définies à partir des distances euclidiennes entre chaque échantillon de la base de données. La parallélisation va consister à calculer ces distances en parallèle. C'est en fait le même processus de parallélisation que l'on retrouve avec les KPPV, où chaque distance est calculée en parallèle. Les auteurs utilisent donc un thread pour calculer une distance. Ils proposent également de calculer ces distances en utilisant une multiplication de matrices car cela est très efficace sur GPU. La distance euclidienne est donc exprimée de manière matricielle. En revanche, cela est difficilement applicable à notre application car les profils distance de test et d'apprentissage sont seuillés à partir de seuils calculés sur chaque nouveau profil distance mesuré. Les matrices d'apprentissage ne seront donc pas exactement les mêmes suivant le profil de test mesuré. En ajoutant également l'étape de recalage, l'utilisation de la forme matricielle de la distance euclidienne n'est pas utilisable dans notre cas. Les résultats présentés par les auteurs

montrent une accélération d'environ 200 de l'algorithme Fuzzy C-means sur GPU par rapport à une implémentation CPU qui n'est en revanche pas détaillée.

En 2012, Ngo et al. (Ngo et al. 2012) ont également traité le même sujet pour une application de classification d'image satellite.

Au vu des articles trouvés dans la littérature, il semble que l'utilisation des GPUs soit une solution particulièrement bien adaptée au domaine de la reconnaissance de cibles où les charges de calcul pour identifier une cible sont souvent très importantes et où l'objectif de traitement en temps-réel est une contrainte forte.

6.3 Algorithme des KPPV et parallélisation sur GPU

Le but de cette section est d'étudier les performances en terme de temps de calcul de l'algorithme des KPPV présenté dans le chapitre 3 (toutes nos implémentations sont réalisées en simple precision, ce qui est suffisant pour notre application). L'étape la plus coûteuse dans l'algorithme des KPPV est l'étape de calcul des distances associée à l'étape de recalage et de seuillage des profils distance. Le reste des calculs est négligeable en terme de temps de calcul (entre 0.3% et 0.4% du temps de calcul total sur CPU).

L'algorithme a été implémenté dans un premier temps dans l'environnement Matlab (version 2012b) de manière vectorisée ou matricielle afin d'optimiser le temps de calcul. Ensuite, différentes étapes d'implémentation sur GPU sont présentées, chacune d'entres elles présentant un niveau d'optimisation de plus en plus poussé. Il va de soit que plus le niveau d'optimisation est important et plus le temps passé pour l'implémenter est important. Nous nous sommes orientés vers une implémentation GPU et nous avons délibérement opté d'étudier en détail les différents points d'optimisation permettant d'améliorer le temps de calcul sur GPU. Les comparaisons en terme de temps de calcul sur CPU (Matlab) et sur GPU sont donc à relativiser car le travail d'optimisation sur CPU et GPU est déséquilibré. Le but est d'obtenir les performances maximum sur un GPU pour atteindre notre objectif de traitement en temps-réel et utiliser au maximum la puissance de calcul offerte par les GPUs.

Dans cette section, après une présentation des opérations nécessaires au calcul des distances, au recalage et au seuillage et des performances dans l'environnement Matlab, nous introduisons les notions importantes à maîtriser (latence, occupation, etc...) lorsque l'on souhaite paralléliser de manière efficace sur GPU. Nous présentons ensuite plusieurs versions implémentées sur GPU, chacune d'entre elles mettant en avant les concepts importants qui ont été utilisés pour améliorer le temps de calcul sur GPU (alignement mémoire, utilisation de la mémoire "shared", parallélisme d'instruction et réutilisation des données). Cette présentation est suivie d'une synthèse des temps de calcul obtenus avec ces différentes versions implémentées en CUDA C++. La section se termine par une comparaison entre les performances obtenues avec CUDA ou OpenCL et une présentation de l'intégration dans l'environnement Matlab des calculs effectués sur GPU en CUDA ou OpenCL (utilisation des mex-files (NVIDIA 2007)).

6.3.1 Nombre d'opérations à virgule flottante

Le nombre d'opérations à virgule flottante (FLO_{KPPV}) nécessaires pour le calcul des distances, le recalage et le seuillage pour un profil distance de test peut s'exprimer de la manière suivante :

$$FLO_{KPPV} = K \times N_A^k \times (2 \times D + 1) \times N \times 3 \tag{6.1}$$

soit en reprenant les dimensions utilisées jusqu'à présent et résumées dans le tableau A.1, environ **0.17 GFLOP** (soit 25 GFLOP pour $N_T = 150$ profils distance de test).

En effet, pour chaque case distance, on effectue 3 opérations à virgule flottante (cf. figure 6.6) pour le calcul de la distance L2 :

- soustraction de l'échantillon de test et d'apprentissage ;
- mise au carré ;
- Accumulation dans la valeur sum.

Le pseudo-code du kernel utilisé pour calculer la distance entre le profil distance de test k et le profil distance d'apprentissage j est exposé sur la figure 6.6.

```
for dec ≤ 2 × D do
   for i ≤ N do
      sum = sum + distance(X_T(k, i + dec), X_A(j, i))
   end for
end for
```

FIG. 6.6 : Pseudo-Code du "kernel"

Les performances pic des GPUs NVIDIA que nous utilisons dans notre étude sont obtenues dans le cas idéal où l'on fait un **fma** (fused multiply add) par cycle. Or les opérations de notre algorithme correspondent en assembleur à un **sub** suivi d'un **fma**. Nous pourrons donc atteindre au mieux que 75% du pic [2].

6.3.2 Temps de calcul sous Matlab

Sous Matlab, nous obtenons un temps de calcul de $1.327s$ (cf. tableau 6.1). Le temps de calcul par profil distance de test est de l'ordre de $1327/150 \approx 9ms$. La contrainte de temps-réel que nous nous sommes fixés pour notre application étant de $1ms$ par profil distance, l'implémentation sous Matlab ne nous permet donc pas d'atteindre cet objectif. A noter que l'utilisation de la Parallel Computing Toolbox (PCT) de Matlab permet de réduire d'un facteur 2 le temps de calcul. Comme on a déjà pu le mentionner dans l'état de l'art, les calculs matriciels sont très efficaces sur GPU. La PCT permet donc aux utilisateurs de Matlab de profiter de la puissance de calcul des GPUs pour réaliser des calculs matriciels qu'ils réalisent habituellement dans l'environnement Matlab sur CPU. La PCT

[2] La puissance pic est obtenue lorsqu'on exécute 1 **fma** par cycle. Un **fma** est compté comme 2 opérations flottantes. Idéalement en 2 cycles on peut faire 2 **fma** donc 4 opérations flottantes. Avec notre algorithme, on est obligé de faire 1 **fma** + 1 **sub** soit 3 opérations flottantes en 2 cycles au lieu de 4 idéalement. D'où les 75% de la puissance pic.

propose donc de charger des matrices sur la carte graphique (via ce qu'on appelle des gpuArray), d'effectuer les calculs sur le GPU puis de récupérer le résultat de ce calcul dans l'environnement Matlab (via la commande gather). Le problème est que tous les traitements ne sont pas réalisables à partir de simples manipulations de matrice (multiplication, addition, etc ...) notamment dans le cadre des traitements NCTR. Dans ce cas la PCT n'est pas suffisamment souple pour s'adapter à ces traitements. Il faut donc se tourner vers une implémentation sur GPU utilisant directement du code CUDA ou OpenCL. C'est ce que nous avons fait dans la suite pour l'algorithme des KPPV car l'accélération obtenue en utilisant la PCT ne permet pas de respecter la contrainte temps-réel.

	Temps(ms)	GFlops
Matlab	1327	25

CPU :Intel Xeon à 2.40GHz

TAB. 6.1 : Temps de calcul (en ms) de l'étape de calcul des distance, recalage et seuillage de l'algorithme des KPPV exécuté en simple precision sous Matlab pour 150 profils de test

6.3.3 Découpage en thread sur l'architecture many-core GPU

L'algorithme des KPPV est un algorithme massivement parallèle. Chaque distance entre un profil distance de test et un profil distance d'apprentissage peut être calculée en parallèle de manière indépendante. L'étape de recalage ne fait qu'augmenter le nombre de distances à calculer et le seuillage ne nécessite que le calcul d'un maximum entre deux floats pour chaque échantillon des profils distance. L'utilisation des GPUs pour améliorer le temps de calcul de l'algorithme des KPPV semble donc être une solution bien adaptée. Pour paralléliser le calcul, on va donc créer $N_{blocks} = 150$ blocs de $N_{threads/block} = 1024$ threads, ce qui correspond aux $N_T = 150$ profils distance de la base de test et aux $N_A = 1024$ profils distance de la base d'apprentissage (nous verrons dans la suite de cette section que le nombre $N_{threads/block}$ peut évoluer de manière à optimiser les performances). Dans le cadre opérationel il n'y aurait qu'un seul profil distance dans la base de test ($N_T = 1$) mais le nombre de profils d'apprentissage serait également beaucoup plus important. Dans ce cas, on pourrait découper la base d'apprentissage en plusieurs blocs pour se ramener à une configuration similaire à celle utilisée pour les données dont nous disposions durant la thèse.

6.3.4 Latence, Occupation et parallélisme

Un des points clés d'une implémentation sur GPU est de "cacher" au maximum la latence d'accès aux ressources de calcul (latence arithmétique) ou aux ressources mémoire (latence mémoire). Il s'agit donc de lancer un maximum d'instructions au cours des temps d'attente.

Latence mémoire et arithmétique

La latence arithmétique est le temps nécessaire pour réaliser une instruction (environ 24 cycles pour un opération arithmétique flottante (Collange 2010)). Il est également courant que plusieurs instructions soient dépendantes les unes des autres (une instruction a besoin du résultat d'une autre instruction pour s'exécuter). Il est donc indispensable, si possible, de "cacher" ces temps d'attente en lançant d'autres instructions (sans dépendance avec les instructions précédemment lancées).

La latence mémoire est le nombre de cycles nécessaires pour accéder à une donnée en mémoire. Cette latence varie en fonction de l'endroit où sont situées ces données. Pour des données en mémoire globale, le nombre de cycles est d'environ 400 à 800 cycles sans cache (Volkov 2010). En utilisant les caches "constante", "texture", L1/L2, la mémoire "shared" ou les registres, plus que quelques cycles sont nécessaires pour accéder aux données.

Accès rapide à la mémoire

Un premier axe d'optimisation consiste à utiliser au maximum les registres puis la mémoire shared et enfin les caches "constante", "texture" ou L1/L2 sur Fermi afin de diminuer la latence mémoire. Par exemple, dans toutes les versions implémentées dans la suite de cette section, les valeurs des seuils appliqués aux profils distance sont stockées en mémoire globale. L'accès à ces données en mémoire globale sera optimisé en utilisant le cache "constante". Dans cette étude, nous n'utilisons pas le cache "texture" car celui-ci n'est pas adapté à notre application (cf. figure 6.7). En effet, nous n'utilisons pas de localité spatiale 2D (pas de réutilisation des données entre deux profils de la base d'apprentissage) pour nos traitements mais nous traitons les données dans une seule dimension, celle du profil distance et de ces cases distance successives. Le cache 2D n'est donc pas intéressant dans ce cas.

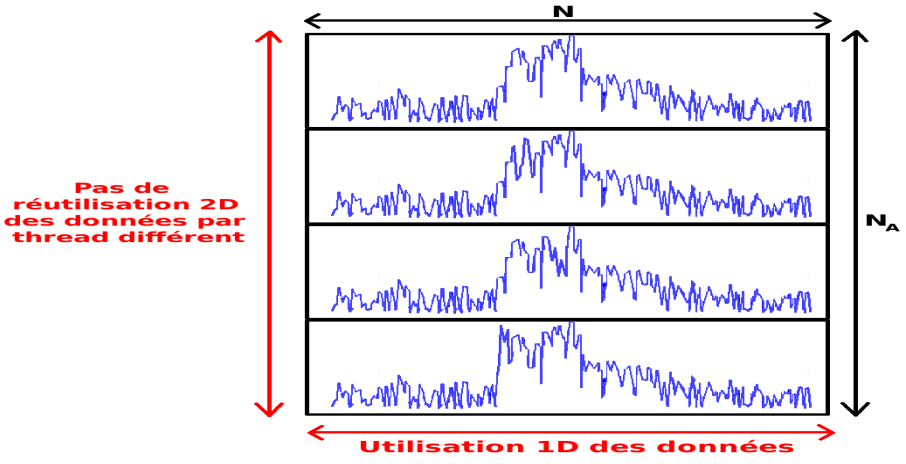


FIG. 6.7 : Utilisation des données

Parallélisme et occupation

Un deuxième axe consiste à "cacher" la latence restante par l'exécution d'autres instructions. Deux démarches principales existent pour traiter cela (Volkov 2010) :

- le parallélisme de threads ou Thread-Level Parallelism (TLP) ;
- le parallélisme d'instructions ou Instruction-Level Parallelism (ILP).

Le but du TLP est d'augmenter l'occupation en lançant le plus de threads de manière à exécuter un maximum d'instructions et à maximiser le débit de calcul.

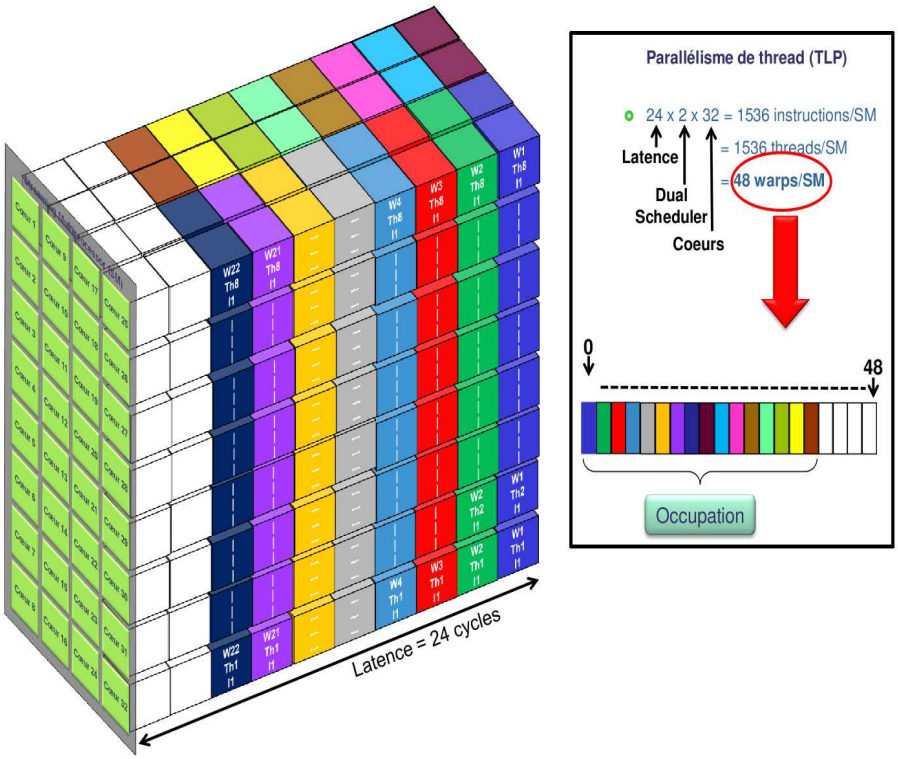


FIG. 6.8 : Parallélisme de threads (inspiré de (Volkov 2010))

La figure 6.8 illustre le principe de parallélisme de threads. Au vu de l'opacité des informations fournies par NVIDIA sur ce sujet, il nous faut tout de même préciser que cette représentation est une interprétation personnelle du mécanisme d'exécution des warps sur GPU et il se peut qu'elle ne colle pas complétement à la réalité. Le cube représente les 32 coeurs d'un Streaming Multiprocessor (SM). Sur chaque coeur, le scheduler va chercher des warps prêts à exécuter une instruction. On considère généralement qu'une instruction est exécutée en 24 cycles (Collange 2010). Le dual scheduler permet de lancer 2 instructions par cycle et chaque SM possède 2×16 coeurs (avec également 16 unités de Load/Store et 4 SFU). On peut donc lancer 1536 instructions/SM, soit 1536 threads/SM soit 48 warps par SM. Le but du TLP est de lancer le maximum de warps sur ces 48 warps maximum possible. Sur ce cube cela consiste donc à occuper le plus de cases possibles, c'est-à-dire de se rapprocher au maximum des 100% d'occupation pour être le plus efficace possible. L'occupation est le

nombre de warps actifs sur le nombre de warps actifs maximum. Avec la carte Fermi, le nombre maximum de warps actifs est de 48 et avec la carte Tesla, ce nombre est de 32. L'occupation est limitée lorsque trop de ressources sont allouées par thread. Il existe quatre facteurs principaux qui peuvent limiter l'occupation :

	Fermi	Tesla
① Nombre maximum de blocs par SM	8	8
② Nombre maximum de warps actifs par SM	48	32
③ Taille maximum de la mémoire registre par SM	32*Ko*	16*Ko*
④ Taille maximum de la mémoire "shared" par SM	48*Ko*	16*Ko*

Tab. 6.2 : Facteurs limitant l'occupation avec les cartes Fermi et Tesla

	Nb threads/bloc (warps/bloc)	1024 (32)	256 (8)	128 (4)
	Mémoire registre/bloc (2Ko registre/thread)	24 Ko	6 Ko	3 Ko
①	Nb max blocs/SM	8	8	8
②	Nb blocs/SM (Nb max de warps actifs/SM)	$\lfloor \frac{48}{32} \rfloor = 1$	$\lfloor \frac{48}{8} \rfloor = 6$	$\lfloor \frac{48}{4} \rfloor = 12$
③	Nb blocs/SM (Taille max mémoire registre/SM)	$\lfloor \frac{32}{24} \rfloor = 1$	$\lfloor \frac{32}{6} \rfloor = 5$	$\lfloor \frac{32}{3} \rfloor = 10$
④	Nb blocs/SM (Taille max mémoire "shared"/SM)	/	/	/
	Min(①,②,③,④) (Warps/SM)	1 (32)	5 (40)	8 (32)
	Occupation	66%	83%	66%

Le symbole $\lfloor\ \rfloor$ représente la division entière

Tab. 6.3 : Calcul de l'occupation en fonction des facteurs limitant pour différentes valeurs de nombre de threads par bloc sur une carte Fermi

L'approche consiste à jouer sur le nombre de threads par bloc pour trouver le meilleur compromis entre les quatre facteurs limitant (cf. tableau 6.2) de manière à lancer un maximum de blocs par SM et à augmenter l'occupation. Le tableau 6.3 résume cela en prenant l'exemple des versions 1 et 2 du tableau 6.4 sur une carte Fermi. Dans ces deux versions, la mémoire "shared" n'est pas utilisée elle n'est donc pas limitante, cependant elle le sera pour les autres versions du tableau 6.4. Un nombre de blocs par SM peut être calculé en tenant compte de chaque facteur limitant. L'occupation est calculée à partir du minimum sur les nombres de blocs par SM obtenus pour chaque facteur limitant. Pour chaque version du tableau 6.4, un tableau de ce type a été construit de manière à trouver le nombre de threads par bloc qui maximise l'occupation.

La deuxième approche pour "cacher" la latence est le parallèlisme d'instructions ou ILP. Il s'agit d'augmenter le nombre d'instructions suc-

cessives sans dépendance de données exécutées par un thread, en réalisant par exemple des déroulages de boucles. A nombre de threads équivalents, l'ILP augmente le nombre d'instructions dans le pipeline. La figure 6.9 illustre le parallélisme d'instructions. Là où le TLP n'avait pas permis une occupation maximale (cases blanches du cube), l'ILP va permettre de lancer une autre instruction dans le pipeline de calcul. Sur la figure 6.9, les warps 1 et 2 vont lancer l'instruction 2 alors que l'instruction 1 n'est pas encore terminée.

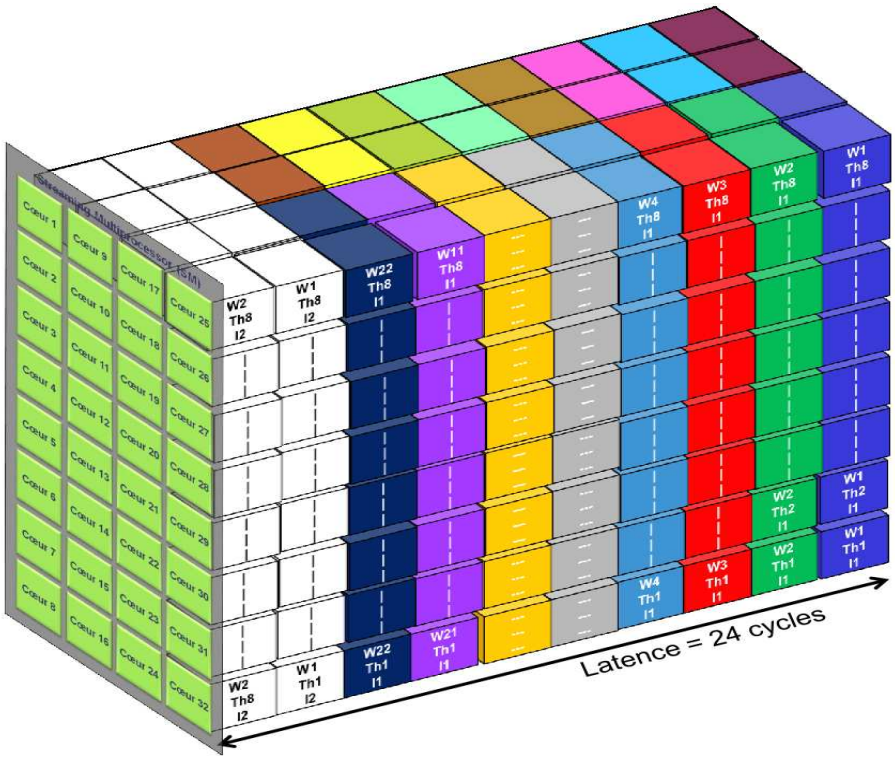


FIG. 6.9 : Parallélisme d'instructions (inspiré de (Volkov 2010))

6.3.5 Implémentation CUDA C++ sur GPU

L'algorithme des KPPV est un algorithme massivement parallèle. Le nombre d'accès mémoire est élevé, il s'agit d'un algorithme "memory bound". Le travail d'optimisation se concentre donc sur les accès mémoire de manière à ce qu'ils soient les plus rapides et efficaces possible. Dans la suite de cette section, plusieurs implémentations (versions 1 à 5 du tableau 6.4) sur GPU des étapes de calcul des distances, recalage et seuillage sont proposées. Les optimisations proposées à chaque version nous permettront de faire ressortir les principales règles de codage à respecter pour obtenir une implémentation efficace sur GPU.

Pour pouvoir analyser les performances des différentes versions, nous avons utilisé les outils de profiling fournis par NVIDIA. Depuis la version 5.0 de CUDA, NVIDIA propose un environnement de développement sous linux nommé "Nsight Eclipse Edition". Il permet entre autres de développer des programmes CUDA, de les débugger et de les "profiler". Le profiler permet d'obtenir un grand nombre d'informations (temps d'exé-

cution des kernels, utilisation de la mémoire, etc...) sur les programmes CUDA exécutés sur GPU. Pour chaque version, nous utilisons ce profiler pour obtenir le temps d'exécution du ou des kernels et le pourcentage d'occupation associé. Ces valeurs sont rassemblées dans les tableaux 6.4 et 6.5.

En plus du profiler inclus dans "Nsight Eclipse Edition", NVIDIA fournit un outil appelé "nvprof" qui permet de collecter les valeurs des compteurs de la carte GPU. Les compteurs qui nous ont intéressés pour notre étude sont les compteurs permettant d'étudier l'accès à la mémoire globale et notamment l'utilisation des caches L1 et L2. Cet outil nous a permis de collecter les valeurs des compteurs du nombre de loads réels (NLR) en mémoire globale, du nombre de lectures en SDRAM (LS), du pourcentage de lecture dans le cache L1 ($\%L1$) et dans le cache L2 ($\%L2$). Toutes ces valeurs sont rassemblées dans le tableau 6.5.

Alignement des données en mémoire : Versions 1 et 2 (V1 et V2)

La version 1 du tableau 6.4 est une implémentation que l'on peut qualifier de "basique". En effet, les profils distance de la base de test et de la base d'apprentissage sont directement stockés en mémoire globale sans se préocuper de l'alignement en mémoire de ces données.

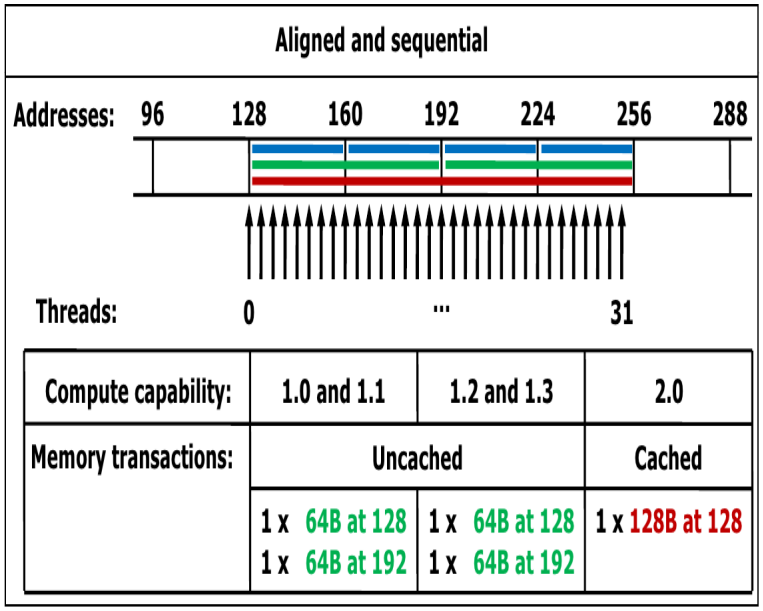


FIG. 6.10 : Alignement et séquentialité des données en mémoire (NVI 2012)

Lorsqu'on stocke des données en mémoire globale, deux points cruciaux sont à respecter (cf. figure 6.10) :

- l'alignement des données ;
- la séquentialité des données : les données utilisées par un même warp doivent être successives.

Ces deux points sont très importants pour faire des accès groupés à la mémoire globale et limiter le nombre de transactions mémoire. La figure

6.11 illustre cela. De manière schématique, on peut voir notre base d'apprentissage comme une matrice de taille $N_A \times N$, N_A étant le nombre de profils distance d'apprentissage et N le nombre de cases distance. Lorsque qu'on mappe cette matrice en mémoire physique profil distance par profil distance (en haut), les threads d'un même warp ne vont pas accéder à des données successives en mémoire. De plus, comme N n'est pas un multiple de 32 la taille d'un warp, les données vont se trouver au fur et à mesure non alignées en mémoire.

En revanche lorsqu'on mappe les données cases distance par cases distance (en bas), les threads d'un même warp vont accéder à des données succesives en mémoire et donc un seul accès en mémoire sera nécessaire pour les 32 threads d'un même warp (1 ligne mémoire de 128 bytes sera chargée).

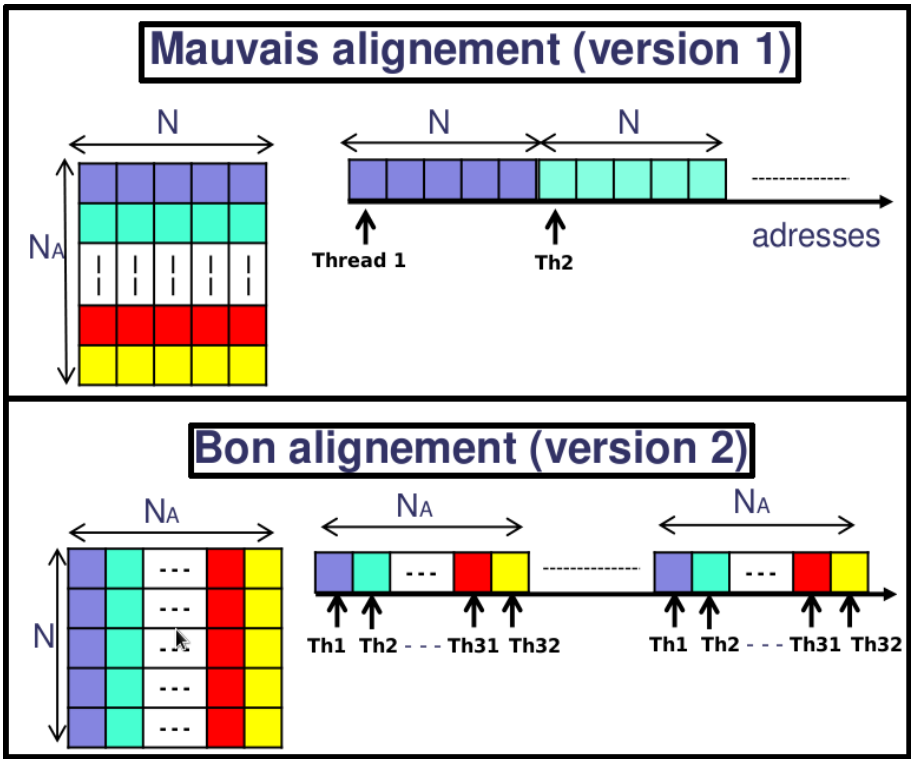


FIG. 6.11 : Stockage en mémoire de la base d'apprentissage

Les performances obtenues avec la première version sont donc très mauvaises que ce soit sur la carte Fermi ou Tesla et bien moins intéressantes que les performances obtenues avec une version Matlab vectorisée. Les défauts du non alignement en mémoire des données sont mis en évidence dans le tableau 6.5. Théoriquement, le nombre de chargements mémoire nécessaires est de 62 Go ($(150 \times 2 \times 50 \times 1024 \times 546)/8/1024^3 \approx 62$ Go). En utilisant, les outils de profiling, on se rend compte qu'avec cette version, 940 Go de chargements mémoire sont en réalité effectués dont une grande partie de lecture directement en mémoire SDRAM (une donnée est lue directement en mémoire SDRAM lorsqu'elle ne se trouve pas dans les caches L1 ou L2). Dans la majorité des cas, chaque thread recharge lui-même la donnée (4 octets) et charge plus de données qu'il en a réellement

besoin (une transaction mémoire charge 128 octets). On atteint seulement 0.36% et 0.84% des performances pics respectivement de la C2050 et de la C1060. L'alignement mémoire est le minimum à respecter pour pouvoir exécuter efficacement des calculs sur GPU.

La version 2 du tableau 6.4 profite quant à elle de l'alignement mémoire. Sur le tableau 6.5, on observe que le nombre de chargements en mémoire a considérablement diminué entre les versions 1 et 2 du fait de l'alignement mémoire. Le simple fait de stocker les données en mémoire de manière intelligente permet d'obtenir un gain important en temps de calcul. Sur la carte Fermi, ce gain est de 28 par rapport à la version 1, ce qui permet d'atteindre 9.9% des performances pic. Le gain est moins important sur la carte Tesla ($\times 8$). Cette différence s'explique en étudiant plus en détail le tableau 6.5. La version 2 sur la carte C2050 profite du cache L2 mis en place sur les cartes de la génération Fermi et non présent sur la carte C1060. Le pourcentage de données demandées se trouvant dans le cache L2 est de 76.9%. Avec la carte C1060, ces données sont chargées directement depuis la mémoire SDRAM sans cache, ce qui augmente considérablement la latence mémoire. Cela accentue la différence en terme de GFlops pic que l'on trouve intrinsèquement entre les deux cartes.

Utilisation de la mémoire "shared" : Version 3 (V3)

Les deux premières versions utilisaient la mémoire globale pour stocker l'ensemble des données. Cependant, l'architecture CUDA propose des moyens pour pouvoir accéder aux données plus rapidement. Un des moyens est de stocker des données en mémoire "shared". La mémoire "shared" est partagée par tous les threads d'un même bloc et peut contenir au maximum $48Ko$ de données avec la génération Fermi et seulement $16Ko$ avec les générations antérieures. L'accès à ces données est très rapide lorsqu'il n'y a pas de conflit de banque entre les threads d'un même bloc.

Lorsque l'on observe l'étape de calcul des distances, on se rend compte que pour les 1024 profils distance de la base d'apprentissage (correspondant aux 1024 threads d'un bloc), on se sert à chaque fois du même profil distance de test. Dans un même bloc, il est donc intéressant de stocker le profil distance de test en mémoire "shared" car tous les threads d'un même bloc y ont accès de manière beaucoup plus rapide qu'en passant par la mémoire globale. Un profil de test contient $N + 2 \times D = 646$ échantillons, ce qui correspond donc à 646 floats soit environ $2.5Ko$. La taille de la mémoire "shared" est donc largement suffisante pour contenir ce profil distance de test que ce soit avec la carte C2050 ou C1060.

La version 3, définie à partir de la version 2 en utilisant de la mémoire "shared" offre un gain de 1.52 pour la carte Fermi et de 1.55 pour la carte Tesla par rapport à la version 2. Cela permet d'atteindre 15.1% des performances pic de la carte C2050 et 10.3% de celles de la carte C1060. Le fait que le nombre de chargements mémoire théoriques a diminué (dans le tableau 6.4, on voit qu'il y a 4Go de lecture en SDRAM en moins) avec cette version (les données de la base de test ne sont plus stockées en mémoire globale, on diminue donc par deux le nombre de chargements nécessaires en mémoire globale) permet d'expliquer cette amélioration. L'utilisation

de la mémoire "shared" bénéficie donc dans les mêmes proportions à la carte C2050 et C1060.

Parallélisme d'instruction (ILP) : Version 4 (V4)

La version 4 reprend les améliorations de la version 3 tout en ajoutant du parallélisme d'instructions (ILP). Un déroulage de boucle de 8 est appliqué sur la boucle des vecteurs d'apprentissage, c'est-à-dire qu'un thread, au lieu de traiter 1 seule distance entre profil distance de test et profil distance d'apprentissage en traite 8. Généralement, le déroulage de boucle est une opération que le compilateur est capable de faire par lui même. Cependant, nous avons observé en étudiant le code assembleur que certaines boucles pouvant être déroulées ne l'étaient pas alors que cela permettrait d'augmenter le parallélisme d'instructions et donc, comme on l'a vu dans le paragraphe 6.3.4, de "cacher" encore mieux la latence mémoire et arithmétique. Cette modification permet d'obtenir 20.1% des performances pic de la carte Fermi et 10.7% de celles de la carte Tesla. On remarque que la gain apporté par le parallélisme d'instructions est plus important pour la carte Fermi que Tesla. Cela s'explique encore en grande partie par la présence des caches L1 et L2 sur la carte Fermi qui n'est pas présent sur la carte Tesla. En observant le tableau 6.5, on se rend compte que très peu de données sont chargées directement en mémoire SDRAM avec la carte Fermi et la majorité des données à charger se trouve déjà dans les caches L1 ou L2. Le gain est faible sur la carte Tesla car, certes, la latence est "cachée" plus efficacement en introduisant de l'ILP mais celle-ci reste importante car les données sont chargées depuis la mémoire SDRAM sans cache.

Réutilisation de données : Version 5 (V5)

En observant le pseudo-code utilisé jusqu'à présent (cf. figure 6.6), on se rend compte que l'on peut encore réduire le nombre de chargements théoriques en mémoire simplement en inversant l'ordre des boucles sur le recalage et les échantillons. En effet, jusqu'à présent pour chaque décalage, on recharge à chaque fois les échantillons du profil d'apprentissage alors que ceux-ci une fois chargés pourraient être réutilisés pour chaque décalage. Le pseudo-code du nouveau kernel CUDA est présenté sur la figure 6.12. Il correspond au kernel utilisé pour la version 5 du tableau 6.4.

```
for i ≤ N do
    for dec ≤ 2 × D/10 do
        sum(j) = sum(j) + distance(X_T(k, i + dec), X_A(j, i))
    end for
end for
```

FIG. 6.12 : Pseudo-Code du "kernel" modifié pour un profil distance de test k et un profil distance d'apprentissage j

En revanche cette inversion de l'ordre des boucles nécessite de stocker en mémoire shared les distances calculées pour chaque décalage (afin de

pouvoir calculer le minimum une fois toutes les distances calculées). Si l'on considère la boucle complète sur les décalages, la taille de la mémoire shared n'est pas suffisante pour stocker l'ensemble des distances calculées. Il faut donc découper la boucle sur les décalages en 10 boucles plus petites. Cela correspond à exécuter 10 fois le même kernel avec un nombre de décalage 10 fois moins grand pour chaque kernel.

La version 5 permet d'augmenter considérablement les performances sur les deux cartes C2050 ou C1060. En effet, le nombre de chargements mémoire théoriques ($3Go$) et effectifs ($2.8Go$) a diminué (cf. tableau 6.5). Par rapport à la version 4, cela représente dix fois moins de chargements mémoire. A noter qu'avec la version 5, l'ILP n'a pas permis d'augmenter les performances. Du fait de la diminution du nombre d'accès mémoire, la latence mémoire semble être entièrement "cachée" par le TLP.

On atteint finalement 40.8% des performances pic de la carte C2050 et 24% de celles de la carte C1060 avec cette dernière version.

Synthèse des performances obtenues

Le tableau 6.4 résume les performances obtenues en terme de temps de calcul pour les différentes versions vues jusqu'à présent. Le tableau 6.5 rassemble quant à lui les caractéristiques des différentes versions implémentées sur GPU. Elles ont été obtenues en utilisant les outils de profiling disponibles sur les cartes graphiques de "compute capabilities" 2.x ou supérieure, ce qui est le cas de la carte C2050 (Nsight + nvprof).

					C2050 (Fermi)			C1060 (Tesla)		
	Al	SM	TLP / ILP	RD	Tps	%P	Gain	Tps	%P	Gain
V1	N	N	TLP	N	6760	0.36		4812	0.84	
V2	O	N	TLP	N	245	9.9	×28	605	6.67	×8
V3	O	O	TLP	N	161	15.1	×1.52	390	10.3	×1.55
V4	O	O	TLP / ILP	N	121	20.1	×1.33	378	10.7	×1.03
V5	O	O	TLP	O	59.7	40.8	×2.03	168	24	×2.25

O=Oui, N=Non, Al=Alignement, SM=Shared Memory, P=Parallélisme, RD=Réutilisation des données, Tps=Temps en millisecondes, %P=Pourcentage de la puissance pic. Le gain calculé pour la version i est calculé relativement à la version $i-1$.

TAB. 6.4 : Propriétés et performances des différentes versions de l'algorithme des KPPV en simple precision sur GPU Fermi et Tesla en CUDA C++

	Occ	NLT	NLR	LS	% L1	% L2
V1	83%	62Go	940Go	763Go	0.74	44
V2	83%	62Go	26.7Go	8.72Go	1.84	76.9
V3	83%	31Go	22.2Go	4Go	5.43	83.7
V4	50%	31Go	11.1Go	0.17Go	25.5	73.2
V5	50%	3Go	2.8Go	1.10Go	0.19	72

Occ=Occupation, NLT=Nombre de load théorique, NLR=Nombre de load réel, LS=Lecture en SDRAM.

TAB. 6.5 : Caractéristiques des différentes versions sur la carte C2050

6.3.6 Implémentation OpenCL

Une implémentation OpenCL de la version 5 a été réalisée de manière à comparer ses performances obtenues avec l'implémentation CUDA. Les résultats sont présentés dans le tableau 6.6.

	C2050 (Fermi)		C1060 (Tesla)	
	Temps(ms)	% Flops pic	Temps(ms)	% Flops pic
V5	85	29%	200	20

TAB. 6.6 : Etape de calcul des distances, recalage et seuillage de l'algorithme des KPPV en simple precision sur GPU Fermi et Tesla en OpenCL

D'une manière générale, les performances obtenues avec l'implémentation OpenCL par rapport à CUDA sont respectivement 30% et 15% moins bonnes sur la C2050 et la C1060 alors qu'il s'agit exactement de la même version que celle implémentée en CUDA. Une première raison à cette différence peut se trouver dans la qualité du compilateur. En étudiant le code assembleur généré par le compilateur OpenCL et CUDA, on se rend compte que certaines optimisations faites par le compilateur CUDA ne sont pas faites par le compilateur OpenCL. L'autre explication peut se trouver dans la qualité des drivers OpenCL et CUDA. Comme l'avait déjà mis en évidence Sawall dans son article de 2011 (Sawall et al. 2011), les drivers OpenCL pour les matériels NVIDIA sont beaucoup moins optimisés que les drivers CUDA.

6.3.7 Implémentation Matlab mex-CUDA et Matlab mex-OpenCL

Lors de la mise au point d'un algorithme, il est intéressant de pouvoir rester dans l'environnement Matlab pour pouvoir tester ces performances tout en profitant de l'accélération apportée par une implémentation sur GPU. Les mex-files permettent d'exécuter du code CUDA ou OpenCL tout en restant dans l'environnement Matlab. Le code de calcul exécuté sous GPU via les mex-files est celui correspondant à la version 5 de l'implémentation sur GPU.

	Temps(ms)	Gain vs Matlab
Matlab	1327	
Matlab CUDA	60	22
Matlab OpenCL	86	15

Tab. 6.7 : Temps de calcul (en ms) de l'étape de calcul des distances, recalage et seuillage de l'algorithme des KPPV exécuté en simple precision sous Matlab, Matlab CUDA et Matlab OpenCL

On retrouve (à quelques millisecondes près) les valeurs obtenues avec les versions CUDA C++ (Version 5 de le tableau 6.4) et OpenCL (Version 5 du tableau 6.7). Il est important de noter que pour obtenir de tels résultats, il est indispensable de séparer les étapes d'initialisation et de remise à zéro du device de l'étape de calcul à proprement parler (exécution du kernel). Ces étapes prennent un temps non négligeable et il est important de ne pas exécuter ces étapes à chaque exécution du mex-file. La procédure est de définir trois mex-files différents (un pour l'initialisation, un pour la remise à zéro et un pour le kernel). Les mex-files d'initialisation et de remise à zero ne sont exécutés qu'une seule fois dans le programme Matlab, ensuite le mex-file correspondant au kernel peut être exécuté autant de fois que l'on souhaite.

6.3.8 Conclusion

L'algorithme des KPPV comme nous l'avons utilisé dans le chapitre 3 est un algorithme "memory bound", c'est-à-dire que les accès à la mémoire sont très nombreux. C'est donc sur l'efficacité de l'accès aux données en mémoire que se sont axées nos optimisations. Une première étude a permis de mettre en évidence l'importance de l'alignement des données en mémoire. Ensuite, à partir de cette implémentation que l'on peut qualifier de "basique" sur GPU, plusieurs améliorations ont été proposées pour augmenter les performances de calcul. Les performances sur un GPU de la génération Fermi et sur un GPU de la génération précédente ont été comparées. Les meilleures performances ont été obtenues sur la carte C2050 de la génération Fermi et permettent de traiter 150 profils distance de test en $59.7ms$ soit $0.40ms$ par profil distance, ce qui est inférieure à la limite que nous nous étions fixés. Cela représente 40% de la puissance pic alors que la puissance maximum atteignable était de 75% de la puissance pic.

En dehors du respect de la contrainte temps-réel, cette étude a permis de mettre en évidence l'intérêt des cartes de la génération Fermi par rapport aux cartes de la génération antérieure. En mettant à part les performances pic plus importantes sur la carte C2050, on se rend compte que les innovations de la génération Fermi (cache L1 et L2, double ordonnanceur de warp) permettent d'exploiter plus facilement et plus rapidemment les puissances de calcul des GPUs. Ces innovations permettent d'atteindre des performances de calcul très convenables avec beaucoup moins d'effort que pour la génération précédente (15.1% des performances pic dès la version 3 pour la carte C2050 alors qu'il faut attendre la version 5 pour atteindre 24% de la performance pic sur la carte C1060). Un gain de 22 a été

obtenu en simple précision entre une version Matlab vectorisée de l'algorithme et une version Matlab CUDA où le calcul des distances, le recalage et le seuillage sont effectués sur GPU. Ce gain est tout même à relativiser à la vue du temps passé beaucoup plus important à optimiser la version GPU. Une implémentation OpenCL a également été réalisée de manière à mettre en lumière les limites de ce standard sur les cartes conçues par NVIDIA au moment où l'étude à été réalisée (CUDA 5.0 et OpenCL 1.2).

Il faut noter que la parallélisation s'est faite jusqu'à présent sur les profils distance de test (N_T) et d'apprentissage (N_A). Dans le cadre opérationnel, on aura généralement qu'un seul profil distance de test dans la base de test et donc la parallélisation ne pourra se faire sur cette boucle. En revanche le nombre de profils distance d'apprentissage sera beaucoup plus important et cela constituera un nouvel axe de parallélisation. Cela ne devrait pas avoir d'impact majeur sur les temps d'exécution car jusqu'à présent seul le nombre de blocs était paramétré par N_T. Au lieu d'avoir un nombre de blocs égal à N_T, on pourrait par exemple découper la base d'apprentissage en plusieurs parties, chacune de ces parties représentant le nombre de blocs de la parallélisation.

6.4 Méthodes probabilistes et parallélisation sur GPU

Théoriquement, l'algorithme basé sur les méthodes probabilistes doit permettre de réduire le coût de calcul par rapport à l'algorithme des KPPV car plutôt que de comparer chaque profil distance entre eux, on compare cette fois-ci uniquement à la moyenne des distributions utilisées en prenant en compte également l'écart-type de ces différentes distributions. Le problème est que les profils distance de la base d'apprentissage doivent être recalés avec chaque nouveau profil distance de test avant l'estimation des paramètres des distributions. Cela passe donc comme avec les KPPV par le calcul des distances entre chaque profil d'apprentissage et le profil de test. Cette étape de calcul des distances reste identique à celle des KPPV, elle peut donc être parallélisée de la même manière que dans la section 6.3. En plus de ce calcul des distances, il faut ajouter en plus l'étape d'estimation des paramètres des K différentes distributions et l'étape de calcul des K probabilités *a posteriori* associées.

Les trois étapes principales de l'algorithme sont donc :

- Etape 1 : Calcul des distances pour recalage ;
- Etape 2 : Estimation des paramètres ;
- Etape 3 : Calcul des probabilités *a posteriori*.

Le pseudo-code de cet algorithme est présenté sur la figure 6.13.

```
for i ≤ N_T do
  for j ≤ K × N_A^k do
    Calcul de la distance (d_i^j) ;
  end for
  for k ≤ K do
    Estimation des paramètres de la distribution de
    Student k ;
  end for
  for k ≤ K do
    Calcul de la probabilité a posteriori d'appartenir à la
    classe k ;
  end for
end for
```

FIG. 6.13 : Pseudo-Code de l'algorithme probabiliste

Coût de calcul global

Le coût global de l'algorithme basé sur les approches probabilistes s'exprime donc à partir des coûts de calcul des trois étapes (FLO_{E1}, FLO_{E2} et FLO_{E3}) détaillés en annexe (cf. annexe A.1.1) :

$$FLO_{PROB} = FLO_{E1} + FLO_{E2} + FLO_{E3} \tag{6.2}$$

Le coût de calcul de ces trois étapes est proportionnel à :

FLO_{E1}	$= O(K \times N_A^k \times (2 \times D + 1) \times N \times 3)$
FLO_{E2}	$= O(K \times N_A^k \times 4 \times N_s^2)$
FLO_{E3}	$= O(K \times 8 \times d_k \times N_s)$

TAB. 6.8 : Coût de calcul pour les trois étapes de l'algorithme probabiliste

En prenant les valeurs définies dans le tableau A.1, on obtient :

$$\mathbf{FLO_{PROB} \approx 0.66\ GFLOP}$$

répartis de la manière suivante :

	GFLOP
FLO_{E1}	0.17
FLO_{E2}	0.16
FLO_{E3}	0.33

TAB. 6.9 : FLOP pour chaque étape de l'algorithme basé sur l'approche probabiliste

L'étape la plus coûteuse est la troisième étape. Les deux premières étapes peuvent être parallélisées suivant le même schéma, c'est-à-dire sur

la boucle sur N_T et la boucle sur $K \times N_A^k$. Il faudrait donc définir deux kernel différents : un pour le calcul des distances (Kernel 1) et un autre pour le calcul des paramètres (Kernel 2). Pour paralléliser la troisième étape, il faudrait paralléliser plutôt suivant la boucle sur N_T et la boucle sur $K \times d_k$. Un troisième kernel (Kernel 3) permettrait de calculer la fonction de coût et la probabilité *a posteriori*. Comme on l'a déjà mentionné avant, dans le cas opérationnel, généralement $N_T = 1$. Par contre, K est beaucoup plus important que 3. On paralléliserait donc plutôt sur les boucles sur K et N_A^k pour les kernel 1 et kernel 2, et sur les boucles sur K et d_k pour le kernel 3.

Pour chacun des kernels, on peut estimer la pourcentage de la puissance pic atteignable en analysant les instructions assembleurs. Néanmoins, il est très difficile d'atteindre ces pourcentages maximum (comme on a pu le voir avec les KPPV où on atteint 40% de la puissance pic alors que la puissance maximum atteignable était de 75%). Généralement, de manière plus réaliste, une implémentation GPU atteignant 25% de la puissance pic est déjà considérée comme étant une bonne implémentation. En se fixant sur ces 25%, on peut estimer les temps de calcul sur les cartes graphique actuelles et du futures.

Le premier kernel doit exécuter **0.17GFlop**. Avec 25% de la puissance pic de la carte C2050 ($257.5GFlops$), ce kernel pourrait être exécuté au mieux en $0.66ms$. Avec le même raisonnement, on obtient un temps d'exécution de $0.62ms$ pour le kernel 2 et de $1.3ms$ pour le kernel 3, soit un total de $2.58ms$ par profil distance. On ne respecte donc pas la contrainte temps réel. A titre indicatif, avec une carte Kepler de dernière génération, c'est-à-dire avec une fréquence pic de $4TFlops$, il faudrait $0.66ms$ par profil distance. En se projetant encore plus loin, avec des cartes GPU de la future génération Volta (prévu en 2016), c'est-à-dire avec une fréquence pic de $6TFlops$ annoncée, on aurait un temps de $0.44ms$ par profil distance. Avec ces deux dernières cartes, la contrainte temps réel est bien respectée.

	Temps (ms)
Génération Tesla	2.58
Génération Kepler	0.66
Génération Volta	0.44

TAB. 6.10 : Temps d'exécution total estimé pour l'algorithme basé sur les méthodes probabilistes ($K = 3$)

Cependant avec un nombre de classes dans la base d'apprentissage plus important, par exemple $K = 30$, le nombre d'opérations à virgule flottante total serait multiplié par 10, soit $\mathbf{FLO_{PROB}} = \mathbf{6.6GFlop}$. En gardant la même structure de parallélisation, le temps d'exécution serait donc 10 fois plus grand. De manière assez simpliste, il faudrait donc environ 25 cartes C2050 pour pouvoir respecter le temps réel. Avec les cartes Kepler, il en faudrait encore 6 alors qu'avec des cartes Volta, 4 cartes pourrait suffire, ce qui devient plus intéressant.

6.5 Algorithme logique floue et parallélisation sur GPU

Nous avons étudié dans le chapitre 5, deux algorithmes basés sur la logique floue. Le but de cette section est d'estimer le coût de calcul de ces deux algorithmes et d'étudier la faisabilité d'une parallélisation sur GPU.

6.5.1 Algorithme D-FRI

L'algorithme D-FRI présenté dans le chapitre 5 n'a pas été implémenté sur GPU. Cet algorithme se découpe en quatres étapes :
- Etape 1 : Calcul des distances ;
- Etape 2 : Calcul des densités de possibilité ;
- Etape 3 : Calcul des possibilités ;
- Etape 4 : Prise de décision.

Les étapes les plus coûteuses sont les trois premières étapes. Comme avec les KPPV, on considère que l'étape de prise de décision est négligeable par rapport aux trois autres. Pour paralléliser, on peut reprendre la structure de parallélisation utilisée pour l'algorithme des KPPV à savoir paralléliser sur les deux premières boucles (boucle sur la base de test et boucle sur la base d'apprentissage) que l'on adaptera pour le cas opérationnel, comme on l'a déjà mentionné avec les KPPV ou les approches probabilistes. Un pseudo-code de cet algorithme est présenté sur la figure 6.14.

for $i \leq N_T$ **do**
  for $j \leq N_A$ **do**
    Calcul de la distance (d_i^j) ;
    Calcul de la densité de possibilité ($DP_{1,i}^j$ et $DP_{2,i}^j$) à partir de d_i^j ;
    Calcul de possibilité (P_i^j) à partir de la densité de possibilité et de la fonction d'appartenance ;
  end for
end for

Fig. 6.14 : Pseudo-Code de l'algorithme D-FRI

Le nombre d'opérations à virgule flottante pour chacune de ces étapes a été évalué en annexe (cf. annexe A.1.2).

Coût de calcul global

Le nombre d'opérations à virgule flottante de l'algorithme D-FRI se résume finalement au coût de calcul de l'étape 1 (FLO_{E1}) car les étapes 2 et 3 ne représentent que très peu d'opérations à virgule flottante en comparaison de l'étape 1 (cf. annexe A.1.2). En effet, le calcul de la densité de possibilité (étape 2) entraîne seulement 2 opérations à virgule flottante et le calcul de la possibilité (étape 3) entraîne dans sa version la moins coûteuse (cf. annexe A.2) qu'une vingtaine d'opérations à virgule flottante.

$$\begin{aligned} FLO_{D-FRI} &= FLO_{E1} \\ FLO_{D-FRI} &= O(K \times N_A^k \times (2 \times D + 1) \times N \times 3) \end{aligned}$$

En reprenant les valeurs du tableau A.1, on obtient donc :

$$\mathbf{FLO_{D-FRI} \approx 0.17\ GFLOP}$$

On peut à nouveau évaluer le temps d'exécution en faisant l'hypothèse que 25% de la puissance pic des cartes GPU utilisées est atteignable. La première étape représente **0.17 GFLOP** donc avec 25% de la puissance pic de la carte C2050 (257.5 GFlops), on peut envisager de traiter cette tâche en **0.66ms**. Les autres étapes étant négligeables, le temps de calcul pour cet algorithme D-FRI est estimé à **0.66ms** par profil distance. Les temps d'exécution avec des cartes de génération Kepler et Volta sont indiqués dans le tableau 6.11.

	Temps (ms)
Génération Tesla	0.66
Génération Kepler	0.17
Génération Volta	0.11

TAB. 6.11 : Temps d'exécution total estimé pour l'algorithme basé sur la logique floue ($K = 3$)

Cette simple estimation nous permet de dire qu'en utilisant de manière efficace les capacités de calcul des GPU, l'objectif temps réel est atteignable avec toutes ces générations de carte GPU. Cependant, le nombre de classes de cibles dans cet exemple est faible, or dans le cas réel ce nombre sera beaucoup plus grand. Cela veut dire que le nombre de profils dans la base d'apprentissage (N_A) sera beaucoup plus important. En multipliant, le nombre de profils distance d'apprentissage par 10 par exemple, on atteindrait un nombre d'opérations à virgule flottante de 1.7 Giga Floating Point Operations. Les temps d'exécution présentés dans 6.11 serait donc multiplié par 10. Pour atteindre le temps réel, il faudrait donc 6 cartes de génération Tesla, entre 1 et 2 de la génération Kepler et 1 seule devrait suffire pour la génération Volta.

6.5.2 Algorithme G-FRI

La structure de l'algorithme G-FRI est différente de celle rencontrée avec l'algorithme D-FRI. On peut déjà dire que le coût de calcul va dépendre de la précision sur la valeur de l'azimut des profils distance de test et d'apprentissage. En effet, plus la précision est grande et plus le nombre de profils distance d'apprentissage et de gabarits à appliquer sur ces profils distance est important (cf. chapitre 5, section 5.2). Le pseudo-code de l'algorithme G-FRI est représenté sur la figure 6.15.

```
for i ≤ N_T do
  Calcul de la densité de possibilité ;
  for k ≤ K do
    for j ∈ Pl_{T,i} do
      for q ∈ Pl_{A,j} do
        Application du gabarit sur j ;
        for dec ≤ 2 × D do
          for N_s = (N > s_D) do
            Calcul de la possibilité ;
          end for
        end for
      end for
    end for
  end for
end for
```

FIG. 6.15 : Pseudo-Code de l'algorithme G-FRI

On identifie trois étapes principales :
- Etape 1 : Calcul de la densité de possibilité ;
- Etape 2 : Application du gabarit ;
- Etape 3 : Calcul de la possibilité.

Coût de calcul global

Le coût global va dépendre du nombre de profils d'apprentissage dans la plage de $Pl_{T,i}$ que l'on note $N_{Pl,i}$ et du nombre de gabarits à appliquer sur ces profils d'apprentissage contenus dans la plage $Pl_{T,j}$ que l'on note $N_{Pl,j}$.

$$\begin{aligned} FLO_{G-FRI} &= FLO_{DP} + K \times N_{Pl,i} \times N_{Pl,j} \times FLO_{Gab} \times (2 \times D + 1) \times N_s \times FLO_{Poss} \\ FLO_{G-FRI} &= O(K \times N_{Pl,i} \times N_{Pl,j} \times 4 \times (2 \times D + 1) \times N_s \times 20) \end{aligned}$$

En reprenant les valeurs du tableau A.1, on obtient :

$$\mathbf{FLO_{G-FRI} \approx 190\ GFLOP} \tag{6.3}$$

En prenant 25% de la puissance pic atteignable, le temps d'exécution de cet algorithme est de $740ms$ avec une carte Fermi. Le tableau 6.12 résume les temps d'exécution estimés avec les trois générations de cartes GPU (Tesla, Kepler et Volta).

	Temps (ms)
Génération Tesla	740
Génération Kepler	190
Génération Volta	126

TAB. 6.12 : Temps d'exécution total estimé pour l'algorithme G-FRI ($K = 3$)

Cet algorithme ne permet pas avec les générations de cartes GPU actuelles ou futures d'atteindre le temps réel. De plus, ce chiffre correspond à une base données de seulement trois cibles. En multipliant par 10, le nombre de cibles dans la base de données, on obtiendrait $\mathbf{FLO_{G-FRI}} \approx \mathbf{1.9}$ **TFLOP**. Autant dire que même en parallélisant des étapes du calcul sur des cartes GPU, il faudrait un grand nombre de cartes GPU pour pouvoir atteindre l'objectif temps-réel.

6.5.3 Accélération de l'étape de calcul des possibilités

Durant notre thèse, nous avons utilisé uniquement la première version de la fonction de calcul de la possibilité (cf. annexe A.2.1). La troisième version (cf. annexe A.2.3) est certes mieux adaptée à une implémentation sur GPU (moins de branchements) mais au vu des coûts de calcul engendrés par l'algorithme G-FRI, nous n'avons pas jugé utile de se lancer dans une parallélisation sur GPU de cet algorithme et donc également de coder cette troisième version de la fonction de calcul de la possibilité. Néanmoins, afin de pouvoir tester les performances de reconnaissance de notre algorithme, nous avons tout de même étudié des solutions permettant d'accélérer simplement et rapidement cette étape de calcul de la possibilité (en utilisant la première version de la fonction de calcul de la possibilité). Le tableau 6.13 résume les temps de calcul de $N_s = 200$ possibilités à partir d'un code Matlab, d'un code C généré via Matlab Coder, d'un code C écrit manuellement, d'un code OpenCL sur CPU.

	Temps(ms)	Gain
Matlab	51.6	
Matlab Coder	4.8	11
C	0.62	83
OpenCL (CPU)	0.23	224

Tab. 6.13 : Temps de calcul (en ms) de 200 possibilités

Matlab Coder

Matlab propose depuis quelques années l'outil Matlab Coder qui permet de générer du code C/C++ autonome à partir d'un code Matlab. Le code source généré est portable et lisible. MATLAB Coder prend en charge un sous-ensemble des fonctionnalités principales du langage MATLAB, en particulier les instructions de condition, les fonctions et les opérations sur les matrices. Il peut générer également les fonctions MEX associées.

L'accélération obtenue par l'intermédiaire de Matlab Coder est intéressante car demandant aucun travail de codage supplémentaire. Néanmoins, elle reste insuffisante pour pouvoir exécuter l'algorithme complet dans un temps raisonnable. Nous nous sommes donc penchés vers le codage de notre propre code C.

Code C

Le code C écrit manuellement pour le calcul des possibilités améliore d'un facteur 7.7 le code C généré par le Matlab Coder et d'un facteur 82 le code Matlab. Encore une fois, le gain obtenu reste insuffisant. Nous nous sommes donc tournés vers d'autres solutions et notamment les solutions multi-coeurs et many-coeurs.

OpenCL

Nous avons choisi l'OpenCL car cela permet de développer un programme exécutable sur GPU ou sur CPU. Le code OpenCL exécuté sur les coeurs du CPU améliore d'un facteur 222 le code Matlab. Cette dernière version OpenCL est celle que nous avons utilisée pour pouvoir accélérer l'algorithme G-FRI basée sur la logique floue. Cette accélération nous a permis de pouvoir tester les performances de notre algorithme en terme de taux d'erreur, taux de succès ou taux de bonne identification.

6.6 Conclusion du chapitre

Une des contraintes fortes pour les applications NCTR est la contrainte de temps-réel (nous nous sommes fixés $1ms$ par profil distance). Dans la section 6.3, l'implémentation de l'algorithme des KPPV sur GPU a été étudiée de manière à respecter cette contrainte. Une étude détaillé a été réalisée pour mettre en avant les points cruciaux à respecter pour se rapprocher au maximum des puissances de calcul des cartes GPU. Dans la section 6.4, nous avons proposé une estimation du coût de calcul de l'algorithme probabiliste présenté dans le chapitre 4. L'implémentation sur GPU de cet algorithme est envisageable et permettrait d'atteindre l'objectifs temps-réel. Nous avons également estimé dans la section 6.5, le coût de calcul des algorithmes basés sur la logique floue. L'algorithme D-FRI présente un coût de calcul de l'ordre de 1.7 GFLOP pour une base d'apprentissage contenant trente types de cibles différentes. Cette charge de calcul est tout à fait absorbable par les cartes GPU actuelles et encore plus par celles à venir. L'algorithme G-FRI est quant à lui beaucoup plus coûteux. Avec trente classes de cibles différentes dans la base d'apprentissage, on atteint près de 1.9 TFLOP. Autant dire qu'avec les cartes GPU actuelles, ce coût de calcul est beaucoup trop important. En se reportant aux spécifications annoncées pour les futures générations de cartes GPU, celles-ci ne permettront également pas de traiter ces charges de calcul en temps-réel. Néanmoins, le coût de calcul a été estimé en prenant des valeurs assez extrêmes. Par exemple, dans l'algorithme actuel, on applique un gabarit tous les $0.05°$ sur l'angle de gisement. En réduisant ce pas à $0.5°$ par exemple, on réduirait le coût de calcul d'un facteur 10. On pourrait faire de même avec les profils d'apprentissage, c'est-à-dire qu'au lieu d'avoir un profil distance d'apprentissage tous les $0.05°$ sur l'angle de gisement, on pourrait en avoir que tous les $0.5°$ par exemple. On réduirait encore le coût de calcul d'un facteur 10. Rien qu'avec ces deux simplifications, le coût de calcul serait raméné à 19 GFLOP, ce qui est une valeur déjà plus accessible pour le temps réel, notamment dans les années futures.

Conclusion générale 7

Dans ce dernier chapitre, nous produisons tout d'abord dans la section 7.1 une synthèse des propriétés des différents algorithmes étudiés durant la thèse et nous revenons sur le comportement de ces différents algorithmes à travers des exemples caractéristiques. Ensuite dans la section 7.2, nous résumons les performances des trois types d'algorithmes proposés au cours de la thèse et nous récapitulons dans la section 7.3 les principaux résultats que l'on a pu obtenir concernant la parallélisation sur GPU. Enfin, nous proposons dans la section 7.4 une synthèse globale des travaux de thèse et nous présentons dans la section 7.5 des perspectives à ces travaux qui pourraient être étudiées de manière à se rapprocher dans les années à venir d'une intégration des traitements NCTR dans les futures générations de radar.

7.1 Synthèse des propriétés des différents algorithmes présentés dans le mémoire

Durant nos travaux de thèse, nous avons pu nous intéresser à trois types d'approches pour faire de la reconnaissance de cibles non coopératives.

- Approche KPPV
- Approche probabiliste
- Approche basée sur la logique floue

Pour pouvoir conclure sur les performances observées avec ces méthodes, il convient tout de même de revenir sur les caractéristiques principales de chacune d'entres elles.

	Nombre de classes acceptées après décision	Modélisation des incertitudes de mesures	Modélisation des données
KPPV	1	Non	Oui
Probabiliste	de 0 à K	Oui	Oui
Logique floue	de 0 à K	Oui	Non

Tab. 7.1 : Propriétés des trois approches étudiées pour nos algorithmes NCTR

Le tableau 7.1 résume les propriétés des trois approches étudiées durant la thèse.

Prise de décision

Le défaut des méthodes de type KPPV est que la décision est prise uniquement parmi les classes présentes dans la base d'apprentissage. Toute décision correspond à accepter une seule classe et donc entraîne le rejet des autres classes. Pour les méthodes probabilites, étant donné que les probabilités d'appartenance à chacune des classes ne sont pas indépendantes, la décision n'est pas prise indépendamment classe par classe. En adaptant la règle de décision et en ajustant les seuils de décision, on peut tout de même faire en sorte d'accepter toutes les classes lorsqu'elles sont équiprobables par exemple, ce qui est intéressant pour la maîtrise du taux d'erreur. Par contre, avec ces méthodes il n'est pas possible de faire une différence entre la situation où toutes les classes doivent être rejetées (profil distance de test appartenant à une classe n'existant pas dans la base d'apprentissage) et la situation où toutes les classes doivent être acceptées car ces deux situations correspondent au cas où les valeurs de probabilité d'appartenance aux K classes sont à peu près égales entre elles. L'avantage des méthodes de type logique floue est que pour chaque classe, on construit indépendamment les unes des autres une fonction d'appartenance. La décision est ensuite prise de manière indépendante pour chaque classe et donc rien ne nous empêche de rejeter toutes les classes ou de les accepter toutes. Cet aspect est très important dans l'objectif du contrôle du taux d'erreur. Lorsqu'on se laisse la liberté de ne pas prendre de décision, on dispose d'un levier pour maîtriser le taux d'erreur.

Avec l'algorithme G-FRI, on est même allé plus loin en prenant une décision case distance par case distance puis en fusionnant ces décisions pour prendre une décision finale. Rétrospectivement, on se rend compte que cela aurait pu également être fait avec les algorithmes de type bayésien. Au lieu de calculer une probabilité *a posteriori* pour un profil distance complet, on aurait pu calculer des probabilités *a posteriori* case distance par case distance et prendre une décision case distance par case distance. Au final, comme avec la logique floue on aurait pu définir le pourcentage de cases distance devant être acceptées pour que le profil distance soit

lui même accepté. En revanche, avec l'algorithme des KPPV, cela semble beaucoup plus compliqué de travailler case distance par case distance.

Modélisation des incertitudes de mesures

Afin de maîtriser le taux d'erreur, il est important de prendre en compte dans nos algorithmes les incertitudes de mesures qui peuvent exister sur les données. Sur ce point, les approches probabilistes ou basées sur la logique floue se placent sur le même plan. La connaissance que l'on a sur la variation de la SER dans une case distance d'un profil distance peut être prise en compte dans les approches probabilistes en utilisant par exemple des modèles de mélange de lois de Student par rapport à des modèles de mélange de Gaussiennes. Avec les algorithmes basés sur la logique floue, cette incertitude est prise en compte dans le calcul de la densité de possibilité. Ces deux approches se distinguent donc sur ce point de l'algorithme des KPPV qui intrasèquement ne permet pas de prendre en compte cette incertitude sur la mesure. Lorsque le RSB du profil distance est favorable (typiquement 30dB), les trois approches donnent des résultats assez similaires. Toutefois, la maîtrise du taux d'erreur se vérifie lorsque le RSB décroit. Ainsi, le taux d'erreur avec les approches de type KPPV augmente fortement pour des RSB faibles (15dB environ) car cette incertitude n'a pas été prise en compte.

Une idée pour prendre en compte ces incertitudes de mesures avec l'algorithme des KPPV serait de définir une nouvelle métrique. La métrique utilisée durant la thèse est la distance euclidienne. En utilisant la distance euclidienne, on fait une hypothèse gaussienne sur la distribution des profils distance. Cette gaussienne a pour matrice de covariance la matrice identité. On pourrait imaginer une métrique qui incorpore la connaissance que l'on a sur la variance de la valeur de la SER case distance par case distance en pondérant, par exemple, la différence de la SER entre deux profils distance dans une case distance par l'inverse de la variance dans cette case distance. En notant v_i, la variance de la SER dans la case distance i, cette métrique est définie de la manière suivante :

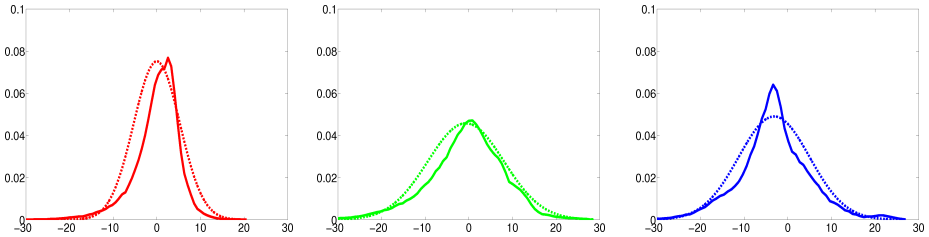
$$d = \sqrt{\sum_i \frac{1}{v_i}(x_i - y_i)^2} \tag{7.1}$$

Cela donnerait donc plus de poids aux cases distance ayant une variance faible et diminuerait l'influence des cases distance dans lesquelles la valeur de la SER varie beaucoup. Dans ce cas, l'hypothèse faite sur la distribution des profils distance resterait gaussienne mais la matrice de covariance serait une matrice diagonale dont les termes diagonaux représente la variance de la SER dans chaque case distance.

Modélisation des données

On peut distinguer l'approche des KPPV et les approches probabilistes de celles basées sur la logique floue par le fait qu'elles modélisent les données. Avec les KPPV, comme on l'a déjà mentionné dans le paragraphe précédent, en utilisant la distance euclidienne on pose une hypothèse gaussienne sur la distribution des profils distance. Idem, avec l'approche

probabiliste, les données sont modélisées par une distribution paramétrique, en l'occurence gaussienne ou de Student (pour prendre en compte l'incertitude sur la mesure). Ces deux distributions ont l'avantage d'être faciles à utiliser. Toutefois, ces lois de distribution ne sont pas toujours très bien représentatives des données traitées. Pour s'en rendre compte, on peut tracer les histogrammes des profils distance de la base d'apprentissage (non bruités) pour chaque classe. Plus précisément, on a tracé sur la figure 7.1, les histogrammes des profils distance sur lesquels on a soustrait le profil distance moyen de chaque classe, de manière à rendre les signaux stationnaires (les courbes en pointillé correspondent aux gaussiennes paramétrées par la moyenne et l'écart-type estimés à partir des histogrammes).



(a) Classe 1 (b) Classe 2 (c) Classe 3

FIG. 7.1 : Histogrammes des profils distance de la base d'apprentissage pour les trois classes.

En observant la forme de ces histogrammes, on se rend compte que l'hypothèse gaussienne n'est pas forcément bien adaptée à nos signaux notamment pour la classe 3. Ces erreurs de modélisation, même très petites, peuvent entraîner une erreur dans la décision finale et donc faire augmenter le taux d'erreur.

L'approche basée sur la logique floue introduit un modèle concernant l'incertitude sur la mesure des profils distance mais n'utilise pas de distributions paramétriques pour modéliser la distribution de ces profils distance. Avec cette approche, on ne fait pas d'hypothèse sur la forme de la répartition des signaux dans les différentes classes. En posant une hypothèse Gaussienne, même si celle-ci est la mieux adaptée à nos données, elle peut ne pas représenter parfaitement les données. Ces erreurs de modélisation, même très petites, peuvent entraîner une erreur dans la décision finale et donc faire augmenter le taux d'erreur.

Comportement des trois algorithmes à travers deux exemples caractéristiques

L'exemple que nous avons pris dans cette section est représentatif du comportement des trois algorithmes. Nous avons volontairement choisi un profil distance d'une classe n'existant pas dans la base d'apprentissage. Le tableau 7.2 résume la classe ou les classes qui ont été acceptées, rejetées ou déclarées incertaines pour chacun des trois algorithmes.

	Acceptation	Incertitude	Rejet
KPPV	2	$\emptyset$	1, 3
Probabiliste	2	$\emptyset$	1, 3
D-FRI	$\emptyset$	$\emptyset$	1, 2, 3

TAB. 7.2 : Résultats de nos trois algorithmes NCTR pour une cible dont la classe n'appartient pas à la base de données

Ce cas caractéristique met en évidence le défaut des algorithmes KPPV et bayésien. Pour les KPPV, la décision est obligatoirement prise parmi les trois classes présentes dans la base d'apprentissage. Ici, une mauvaise classe est acceptée et une information erronée est transmise à l'opérateur. Dans le cas bayésien, il aurait fallu modéliser en plus des trois classes présentes dans la base d'apprentissage, une classe "autre". Mais trouver un modèle représentant des classes de cibles très différentes les unes des autres et pour lesquelles aucun échantillon d'apprentissage n'est disponible est très complexe et difficile à mettre en oeuvre (il existe dans la littérature (Rasmussen 2000) des modèles appelés "Infinite GMM" qui peuvent éventuellement permettre de traiter ce cas spécifique). Comme avec les KPPV, on va donc prendre une décision parmi les trois classes présentes dans le modèle de mélange. Et comme avec les KPPV, une mauvaise classe est acceptée et la réaction de l'opérateur risque d'être inadaptée à la situation. Avec l'algorithme basé sur la logique floue, on s'autorise à rejeter les trois classes et dans ce cas, c'est ce qui se produit. Ce comportement est beaucoup plus en adéquation avec le contexte NCTR.

On peut étudier également le cas où, cette fois-ci, le profil distance de test à reconnaître est un profil distance de test de la classe 1 assez bruité donc étant assez éloigné des profils distance de la base d'apprentissage. Les décisions prises pour chacun des algorithmes sont résumées dans le tableau 7.3.

	Acceptation	Incertitude	Rejet
KPPV	3	$\emptyset$	1, 2
Probabiliste	2	$\emptyset$	1, 3
D-FRI	1, 2, 3	$\emptyset$	$\emptyset$

TAB. 7.3 : Résultats de nos trois algorithmes NCTR pour un profil distance de la classe 1 fortement bruité

Pour l'algorithme des KPPV, on cherche les distances les plus faibles parmi les distances entre le profil de distance de test et les profils distance d'apprentissage. Dans ce cas, la classe qui revient le plus souvent parmi les K plus proches voisins est la classe 3. En fait, le profil distance de test étant bruité, on retrouve des cases distance caractéristiques de la classe 1 sous le niveau de bruit et donc seuillées. Lorsque l'on calcule la distance, ces cases distances ne sont pas prises en compte, ce qui fausse la mesure de distance. La fluctuation possible de la valeur de la SER dans une case distance n'est pas prise en compte, il y a donc un risque accru de faire

des erreurs et c'est ce qui se passe sur l'exemple exposé dans cette section. D'autant plus que seule une classe est retenue avec la règle de décision des KPPV.

Pour l'algorithme basé sur l'approche probabiliste, la fluctuation de la valeur de la SER dans une case distance est prise en compte dans le modèle. En revanche la valeur de la probabilité *a posteriori* calculée pour une classe n'est pas indépendante des autres classes. Dans ce cas, où le profil distance de test est plutôt éloigné des profils distance d'apprentissage, les distances entre celui-ci et le modèle de chaque classe sont plutôt grandes. Il suffit que le profil distance de test soit un peu plus proche d'une des classes (même avec une distance assez grande) pour que celle-ci soit favorisée par rapport aux deux autres. De plus, contrairement à l'approche des KPPV ou à l'approche logique floue, on ne compare pas le profil distance de test à tous les profils distance d'apprentissage mais seulement à la moyenne des profils distance d'apprentissage (en tenant compte de l'écart-type et de la fluctuation de la mesure de la SER dans une case distance). Pour toutes ces raisons, la probabilité d'appartenir à la classe 2 est élevée et même en ajustant la règle de décision, cela ne nous permet pas de ne pas commettre d'erreur.

En revanche, pour l'algorithme basé sur la logique floue, les trois classes sont retenues car l'indécision est trop forte. Contrairement au cas probabiliste, chaque valeur de distances entre le profil distance de test et les profils d'apprentissage est comparée avec les fonctions d'appartenance de chaque classe (calculées à partir des profils distance de la base d'apprentissage). En construisant les densités de possibilités à partir de cette valeur de distance, on prend en compte l'incertitude sur la mesure de la SER dans une case distance. Les possibilités calculées sont indépendantes classes par classes. La décision prise pour une classe est donc indépendante de celles prises pour les autres classes. Plutôt que de faire une erreur et de prendre une décision à tout prix, on préfère retenir les trois classes et s'assurer un taux d'erreur faible.

7.2 Synthèse des performances des différents algorithmes présentés dans le mémoire

Durant cette thèse, nous nous sommes intéressés à trois types d'algorithmes pour faire de la reconnaissance de cibles non coopératives. Le premier algorithme basé sur l'algorithme des KPPV est un algorithme qui permet d'obtenir de bons résultats sur des données synthétiques mais dès que le RSB devient trop faible, le taux d'erreur devient trop important. Le deuxième type d'algorithme étudié est un algorithme basé sur les méthodes probabilistes. Cet algorithme présente comme avantage par rapport à l'algorithme des KPPV de fournir une mesure de confiance quant à la décision prise. Un algorithme basé sur des mélanges de lois de Student a été présenté et étudié. Les résultats obtenus n'ont pas été satisfaisants car même en adaptant la règle de décision pour tenter de maîtriser le taux d'erreur, celui-ci devient toujours beaucoup trop élevé lorsque le RSB diminue. La dernière famille d'algorithme étudiée est celle basée sur la logique floue. La logique floue offre des mécanismes simples et bien adaptés

au contexte de la reconnaissance de cibles non coopératives. Un premier algorithme (algorithme D-FRI) a permis de mettre en évidence l'efficacité de ce type d'algorithme pour ce qui est de la maîtrise du taux d'erreur par rapport aux deux algorithmes précédents (cf. tableau 7.4). En revanche sur des données réelles, l'algorithme D-FRI permet toujours de contrôler le taux d'erreur mais donne des résultats en terme de taux de succès beaucoup trop faibles (cf. tableau 7.5). Un second algorithme basé sur la logique floue (algorithme G-FRI) et intégrant des données mesurées en chambre sourde a donc été proposé dans ce mémoire. Celui-ci permet d'améliorer les performances en terme de taux de bonne identification et de succès tout en gardant la maîtrise du taux d'erreur (cf. tableau 7.5).

		TE	TBI	TS
KPPV	RSB=30dB	0	100	100
	RSB=20dB	6.6	93.4	93.4
	RSB=15dB	14	86	86
Probabiliste	RSB=30dB	0	70.67	100
	RSB=20dB	6	91.3	94.0
	RSB=15dB	18	70.7	74.7
D-FRI	RSB=30dB	0.6	95.3	96.0
	RSB=20dB	1.3	45.3	49.3
	RSB=15dB	2	10.67	17.33

Tab. 7.4 : Taux d'erreur, de bonne identification et de succès obtenues avec les trois différents types d'algorithmes étudiés durant la thèse et sur les données synthétiques

		TE	TBI	TS
D-FRI	Données réelles	0	0	0
G-FRI	Données réelles	4.0	30.1	37.5

Tab. 7.5 : Taux d'erreur, taux de succès et taux de bonne identification pour les données réelles et pour les deux algorithmes basés sur la logique floue.

7.3 Synthèse des performances de parallélisation

Nous avons étudié la parallélisation sur GPU des algorithmes de reconnaissance de cibles non coopératives. Nous avons présenté dans ce mémoire une étude détaillée de la parallélisation de l'algorithme basé sur les KPPV sur un processeur "many-core" de type GPU. A travers cette étude, nous avons mis en évidence les moyens à mettre en oeuvre pour paralléliser le plus efficacement possible nos algorithmes NCTR sur GPU. Le principal effort d'adaptation de ces algorithmes "memory bound", limités par les nombreux accès mémoire, a consisté à utiliser de manière pertinente les registres, la mémoire "shared", la mémoire cache en réor-

donnant la séquence de calculs pour créer la séquence d'accès mémoire la moins coûteuse en temps d'accès.

Les autres algorithmes n'ont pas été entièrement parallélisés sur GPU. Pour l'algorithme basé sur le mélange de lois de Student, nous avons évalué brièvement la charge de calcul éventuelle d'une parallélisation sur GPU mais les performances de reconnaissance de l'algorithme n'ont pas justifié une étude plus approfondie de son implémentation. Le raisonnement a été le même pour l'algorithme D-FRI basé sur la logique floue qui ne donnait pas des résultats suffisamment intéressants pour que l'on passe du temps sur une parallélisation sur GPU. Pour l'algorithme G-FRI, le coût de calcul étant très supérieur à l'algorithme D-FRI, nous avons du nous pencher sur un moyen d'accélérer le code pour pouvoir simplement tester les performances de notre algorithme.

Au final, les travaux de thèse ont validé l'intérêt d'une implémentation sur GPU des algorithmes de reconnaissance des profils distance, du fait de leurs natures hautement parallélisables et cela malgré la nature "memory bound" de ces algorithmes.

7.4 Synthèse globale

Les travaux menés au cours de cette thèse démontrent l'intérêt d'une implémentation sur GPU des algorithmes de reconnaissance de cibles non coopératives basés sur l'exploitation du profil distance. En effet, les algorithmes proposés sont fortement parallélisables (comme par exemple sur les cases distance à traiter, sur le nombre de classes dans la base d'apprentissage, ...), pour un faible débit de données entrant vers le GPU : le profil distance mesuré. Il est également utile de rappeler que les profils distance formant la base d'apprentissage sont connus avec un niveau de fiabilité pouvant varier en fonction de leur origine (mesure ou simulé) :

- pour les profils distance mesurés, à niveau de bruit équivalent et à présentation (distance, azimut, site) équivalente, ceux-ci peuvent fluctuer, par exemple à cause de la rotation des moteurs ;
- pour les profils distance synthétiques, on retrouve la fluctuation engendrée par la rotation des moteurs. A cela se rajoute le degré de représentativité des calculs de la SER par simulation. Par exemple, la réflexion des ondes électromagnétiques au sein des conduits d'air est un phénomène complexe à modéliser. Par conséquent, au niveau d'un conduit d'air, le degré de représentativité d'un profil distance simulé est moindre.

Il est donc intéressant de prendre en compte explicitement dans nos algorithmes ce degré de fiabilité des profils distance formant la base d'apprentissage. Parmi les quatre algorithmes proposés, deux le permettent :

- l'algorithme G-FRI, avec l'exploitation de gabarits par angle d'incidence. Ainsi, le gabarit sera large pour les zones de l'avion à incertitude élevée (conduit d'air, queue de l'appareil, moteur, ...) et fin pour les zones à incertitude faible (nez de l'appareil, cockpit, ...) ;
- l'algorithme basé sur les approches probabilistes, à l'aide du degré de liberté de la loi de Student.

Ces deux algorithmes permettent ainsi une amélioration progressive des performances de classification par l'introduction de profils distance de plus en plus fiables dans la base d'apprentissage, sans avoir à modifier la nature des traitements. Ils sont donc les plus à mêmes à être intégrés dans les radars :

- l'algorithme probabiliste implémenté à ce jour offre une charge de calcul raisonnable, mais avec un taux d'erreur faible uniquement pour des RSB élevés ;
- l'algorithme G-FRI présente une charge de calcul très importante, mais avec maîtrise du taux d'erreur. Cette charge de calcul peut être réduite en "comprimant" les données de la base d'apprentissage (réduction du pas d'échantillonnage en gisement et en site par exemple), mais ceci se faisant au détriment du taux de succès à taux d'erreur constant.

7.5 Perspectives

Parmi les algorithmes étudiés au cours de la thèse, seul l'algorithme G-FRI permet actuellement de s'approcher des contraintes imposées par les problématiques NCTR :

- maîtrise du taux d'erreur ;
- maximisation du taux de succès ;
- calcul en temps réel.

Pour que l'on puisse envisager une utilisation de cet algorithme dans un contexte opérationnel, il faudrait néanmoins le soumettre à d'autres tests notamment en augmentant le nombre de cibles possibles à identifier et en faisant varier le niveau de RSB dans notre base de données. D'un point de vue implémentation, la thèse a montré que les GPUs étaient une voie intéressante (qu'il faudrait bien entendu approfondir dans la perspective d'une implémentation opérationnelle) pour compenser la contrainte de temps de calcul des algorithmes G-FRI. A cet effet, il serait intéressant d'envisager dans un avenir proche une parallélisation sur les nouvelles génération Kepler de cartes GPU puis à moyen terme une parallélisation sur des cartes de la génération Volta prévue à l'horizon 2016. Il faudrait notamment évaluer l'impact d'une réduction de la précision sur les angles de gisement sur les performances de reconnaissance, ce qui permettrait de réduire les coûts de calcul et par la même occasion de se rapprocher de l'objectif temps réel (ou même l'atteindre) en parallélisant sur les cartes GPU. D'une manière générale, on peut envisager à moyen terme des solutions opérationnelles du même type que l'algorithme G-FRI combinées, par exemple, à des représentations multirésolutions qui pourraient nous permettre d'adapter de manière efficace le dégré de raffinement de nos traitements à la puissance de calcul maximale offerte par les cartes GPU. A long terme, on peut considérer que les technologies GPU, nous permettront d'atteindre le degré de raffinement optimum tout en respectant la contrainte temps-réel.

A moyen terme, horizon 2025, en se placant dans une logique d'intégration dans un produit industriel, l'algorithme G-FRI apparaît comme

étant la solution la mieux adapté et la plus efficace. En effet, les principes de la logique floue s'accordent idéalement à la problématique particulière des applications NCTR, à savoir la maîtrise du taux d'erreur. En revanche, au vu des résultats obtenus avec l'algorithme G-FRI et de l'utilité de l'information introduite dans cet algorithme (gabarits obtenus en chambre sourde), il serait également intéressant, à plus long terme, de revenir sur les approches KPPV et probabilistes pour étudier les moyens d'introduire cette information supplémentaire avec ces deux types de méthode et d'étudier les performances obtenues en intégrant ces informations supplémentaires. Avec les méthodes probabilistes, une idée serait de définir des classes pour les différents types de cible mais également des sous-classes à l'intérieur de ces classes pour modéliser la valeur de la SER case distance par case distance. Le modèle choisi pour ces sous-classes devra intégrer l'information fournie par les gabarits obtenus en chambre sourde. Toujours avec les méthodes probabilistes, il serait également intéressant d'étudier les modèles dit "Infinite GMM" qui pourraient permettre de traiter le cas où les profils distance de test appartiennent à une classe non présente dans la base d'apprentissage.

Un autre axe de recherche serait d'étudier l'adaptation des solutions proposées dans le cadre de la thèse au monde de l'imagerie 2D. En restant dans le domaine des applications NCTR, la technique ISAR 2D, par exemple, permet de passer de l'univers 1D des profils distance à une représentation 2D des cibles. Les résultats de cette thèse pourraient être déclinés à la reconnaissance ISAR 2D et pourraient donner aux travaux de recherche actuelles une autre orientation tout aussi pertinente. D'une manière générale, on pourrait élargir le domaine d'application des solutions présentées durant la thèse au domaine de l'imagerie spatiale, dans lequel on retrouve des problématiques assez proches de celles évoquées durant la thèse.

Annexes

A

Sommaire

A.1 Estimation des coûts de calcul 150
A.1.1 Méthodes probabilistes 150
A.1.2 Algorithme D-FRI 152
A.1.3 Algorithme G-FRI 153
A.2 Fonctions de calcul de la possibilité 154
A.2.1 Pseudo-code de la première version de la fonction de calcul de la possibilité 154
A.2.2 Pseudo-code de la deuxième version de la fonction de calcul de la possibilité 156
A.2.3 Pseudo-code de la troisième version de la fonction de calcul de la possibilité 157
A.3 Publications liées à la thèse 160

A.1 Estimation des coûts de calcul

Nombre de classes de cibles	K	3
Nombre de profils distance dans la base d'apprentissage pour la classe k	N_A^k	342
Décalage	D	50
Nombre de cases distance	N	546
Nombre de cases distance au dessus du seuil au bruit	N_s	200
Dimension intrasèque du sous-espace de la classe k	d_k	200
Nombre de points d'interpolation pour le calcul de la possibilité	N_I	10000
Nombre de profils distance sélectionnés par classe pour faire la reconnaissance avec l'algorithme G-FRI	$N_{Pl,i}$	200
Nombre de gabarits à appliquer aux profils distance d'apprentissage	$N_{Pl,j}$	200

Tab. A.1 : Valeurs des variables pour l'estimation des coûts de calcul.

La valeur de N_{Pl} va dépendre de la précision sur la mesure de l'azimut des profils distance. En fixant cette précision à 5° et en considérant que dans la base d'apprentissage, on a un profil distance tous les 0.05° donc la taille de la plage $Pl_{T,i}$ est de $N_{Pl,i} = 200$. On applique le même raisonnement pour $N_{Pl,j}$.

A.1.1 Méthodes probabilistes

Etape 1

Le coût de calcul pour l'étape 1 a déjà été évalué dans la section 6.3.

$$\begin{aligned} FLO_{E1} &= K \times N_A^k \times FLO_{Dist} \\ FLO_{E1} &= O(K \times N_A^k \times (2 \times D + 1) \times N \times 3) \end{aligned}$$

Etape 2

L'estimation des paramètres consiste à calculer la moyenne, la matrice de covariance et le degré de liberté ν de chaque loi de Student (cf. section 4.1.2). Le degré de liberté est estimé en amont car il ne dépend pas de la base d'apprentissage.

Le calcul de la moyenne consiste simplement à faire une somme sur tous les profils distance d'apprentissage dans toutes les cases distance et de diviser par le nombre total de profils distance d'apprentissage. Il faut prendre en compte également que les profils distance sont seuillés en fonction du bruit mesuré sur le profil distance de test. Cette étape 2 ne peut donc pas se faire en amont car pour chaque nouveau profil distance de

test, on doit réestimer les paramètres du modèle. Certaines cases distance sont situées en-dessous du seuil, elles ne sont donc pas prises en compte. Le nombre de cases distance au-dessus du seuil dépend du niveau de bruit du profil distance de test mesuré. On note N_s, le nombre moyen de cases distance au-dessus du seuil.

$$FLO_{Moyenne} = N_A^k \times N_s$$

Les matrices de covariance sont de taille $N_s \times N_s$. Chaque terme est calculé de la manière suivante :

$$\boldsymbol{C}_A^k(p,q) = \frac{1}{N_A^k - 1} \sum_{i=1}^{N_A^k} (x_{A,i}^k(p) - m_A^k(p))(x_{A,i}^k(q) - m_A^k(q))$$

Le nombre d'opérations à virgule flottante pour le calcul de la matrice de covariance est donc :

$$\begin{aligned} FLO_{Covariance} &= N_s \times N_s \times (1 + (N_A^k - 1) + N_A^k \times (2+1)) \\ &= N_s \times N_s \times 4 \times N_A^k \end{aligned}$$

En effet, le calcul de chaque terme de la matrice nécessite N_A^k multiplications, $N_A^k - 1$ sommes, $2 \times N_A^k$ soustractions et 1 division.

Le nombre d'opérations à virgule flottante pour l'étape 2 est donc :

$$\begin{aligned} FLO_{E2} &= K \times (FLO_{Moyenne} + FLO_{Covariance}) \\ FLO_{E2} &= K \times N_A^k \times N_s \times (1 + 4 \times N_s) \\ FLO_{E2} &= O(K \times N_A^k \times 4 \times N_s^2) \end{aligned}$$

Etape 3

En se reportant à la section 4.2 du chapitre 4, on se rend compte que cette étape se résume aux calculs des K fonctions de coût associées aux différentes classes du modèle. On rappelle que la fonction de coût peut être exprimée ainsi :

$$\begin{aligned} R_k(\boldsymbol{x}_{T,i}) &= ||\boldsymbol{m}_A^k - H_k(\boldsymbol{x}_{T,i})||_{A_k}^2 + \frac{1}{b_k}||\boldsymbol{x}_{T,i} - H_k(\boldsymbol{x}_{T,i})||^2 + \sum_{j=1}^{d_k} \log(a_{kj}) \\ &\quad + (d - d_k)\log(b_k) - 2\log(\pi_k) + d\log(2\pi) \end{aligned}$$

et que la probabilité *a posteriori* est déduite de R_k de la manière suivante :

$$P_k = \frac{\exp(-\frac{1}{2}R_k)}{\sum_{k=1}^{K} \exp(-\frac{1}{2}R_k)} \tag{A.1}$$

Le calcul de la fonction de coût R_k nécessite des multiplications vecteur-matrice ou matrice-matrice. Elle se décompose en plusieurs étapes :

- Calcul de $\boldsymbol{m}_A^k - H_k(\boldsymbol{x}_{T,i})$: cette étape nécessite $4 \times N_s \times d_k + N_s$ opérations à virgule flottante ;
- Calcul de $||\boldsymbol{m}_A^k - H_k(\boldsymbol{x}_{T,i})||_{A_k}^2$: cette étape nécessite $2 \times (d_k \times (d_k + 2 \times N_s) + N_s)$ opérations à virgule flottante ;
- Calcul de $\frac{1}{b_k}||\boldsymbol{x}_{T,i} - H_k(\boldsymbol{x}_{T,i})||^2$: cette étape nécessite $4 \times N_s + 1$ opérations à virgule flottante ;
- Calcul de $\sum_{j=1}^{d_k} \log(a_{kj})$: cette étape nécessite $2 \times d_k$ opérations à virgule flottante, en considérant 1 opération à virgule flottante pour le calcul du logarithme ;
- Calcul de $(p - d_k)\log(b_k) - 2\log(\pi_k) + p\log(2\pi)$: cette étape est négligeable par rapport aux étapes précédentes.

Le nombre d'opérations à virgule flottante pour le calcul de R_k est donc :

$$\begin{aligned} FLO_{Fcout} &= 4 \times N_s \times d_k + 2 \times (d_k \times (d_k + 2 \times N_s) + N_s) + 5 \times N_s + 1 + 2 \times d_k \\ &= 4 \times N_s \times d_k + 4 \times N_s \times d_k + 2 \times d_k^2 + 5 \times N_s + 2 \times d_k + 1 \\ FLO_{Fcout} &= O(8 \times N_s \times d_k + 2 \times d_k^2) \end{aligned}$$

D'après l'équation A.1, le calcul de la probabilité *a posteriori* consiste à multiplier la fonction de coût par 1/2, à calculer l'exponentielle des K fonctions de coût et au final à diviser par la somme. On peut considérer que le calcul de l'exponentielle correspond à 1 opération à virgule flottante. Le nombre d'opérations à virgule flottante pour le calcul de la probabilité *a posteriori* est donc :

$$\begin{aligned} FLO_{Post} &= K \times (1 + 1) + (K - 1) + 1 \\ &= 3 \times K \end{aligned}$$

Au final, le coût de calcul pour cette étape est de :

$$\begin{aligned} FLO_{E3} &= K \times (FLO_{Fcout}) + FLO_{Post} \\ FLO_{E3} &= O(K \times 8 \times N_s \times d_k + 2 \times d_k^2) \end{aligned}$$

A.1.2 Algorithme D-FRI

Etape 1 : Calcul des distances

Comme on a pu le voir avec l'algorithme des KPPV, le calcul des distances inclut une étape de recalage puis pour chaque échantillon, on exécute 3 opérations (Soustraction, Mise au carré, Accumulation).

$$FLO_{E1} = O(K \times N_A^k \times (2 \times D + 1) \times N \times 3)$$

Etape 2 : Calcul des densités de possibilités

L'étape de calcul des densités est beaucoup plus simple car elle consiste à calculer deux grandeurs à partir de la valeur de la distance

et de deux quantiles. Pour la première grandeur, on soustrait à la valeur de la distance, la valeur du premier quantile et pour la deuxième grandeur la valeur du second. On exécute donc deux soustractions.

$$FLO_{E2} = O(K \times N_A^k \times 2)$$

Etape 3 : Calcul des possibilités

Cette étape est détaillée dans la section A.2. En fonction des versions utilisées, le coût de calcul varie beaucoup. Dans l'optique d'une parallélisation sur GPU, la troisième version semble la mieux adaptée. Le nombre d'opérations à virgule flottante pour cette troisième version du calcul de la possibilité se résume à une vingtaine d'opérations à virgule flottante. Le coût de calcul pour cette troisième étape peut donc être exprimé de la manière suivante :

$$\begin{aligned} FLO_{E3} &= O(K \times N_A^k \times FLO_{Poss}) \\ FLO_{E3} &= O(K \times N_A^k \times 20) \end{aligned}$$

A.1.3 Algorithme G-FRI

Etape 1 : Calcul de la densité de possibilité

Contrairement à la première version de l'algorithme, la densité de possibilité est composée de quatre grandeurs. Dans cette deuxième version de l'algorithme, la densité de possibilité est symétrique et centrée autour de la valeur de la SER dans une case distance. Cela n'était pas le cas avec la première version de l'algorithme car la densité de possibilité est calculée sur la mesure de distance et cette mesure ne peut pas être sous-estimée. Le nombre d'opérations à virgule flottante pour cette étape est donc égale à 2 soustractions et 2 additions.

$$FLO_{DP} = 4$$

Etape 2 : Application du gabarit

Cette étape consiste à construire autour de la valeur de la SER un gabarit. Si on se rapporte à la section 5.2.1, cela correspond donc à 4 opérations à virgule flottante (3 soustractions et 1 addition).

$$FLO_{Gab} = 4$$

Etape 3 : Calcul de la possibilité

On reprend le nombre d'opérations à virgule flottante calculé pour l'algorithme D-FRI :

$$FLO_{Poss} = 20$$

A.2 Fonctions de calcul de la possibilité

A.2.1 Pseudo-code de la première version de la fonction de calcul de la possibilité

La fonction de calcul de la possibilité prend comme paramètres d'entrée 2 matrices (une pour la fonction d'appartenance et une pour la densité de possibilité) de taille 4×2. La première ligne de la matrice correspond aux coordonnées sur l'axe des abscisses et la deuxième ligne aux coordonnées sur l'axe des ordonnées. Soit FA et DP ces deux matrices :

$$FA = \begin{pmatrix} FA_{x,1} & FA_{x,2} & FA_{x,3} & FA_{x,4} \\ FA_{y,1} & FA_{y,2} & FA_{y,3} & FA_{y,4} \end{pmatrix}, DP = \begin{pmatrix} DP_{x,1} & DP_{x,2} & DP_{x,3} & DP_{x,4} \\ DP_{y,1} & DP_{y,2} & DP_{y,3} & DP_{y,4} \end{pmatrix}$$

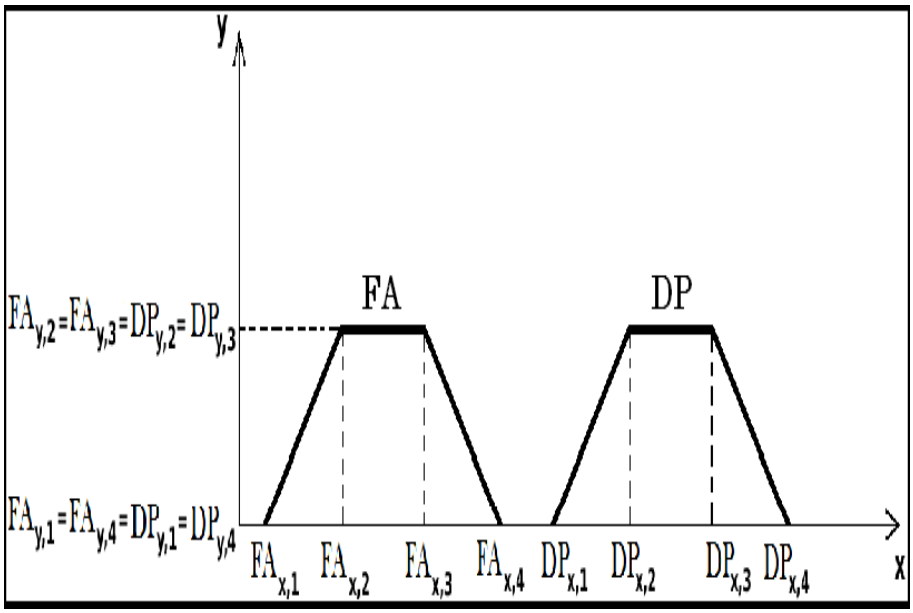


Fig. A.1 : Fonction d'appartenance et densité de possibilité

P = Possibility1(FA,DP) :
mi = min($FA_{x,1}, DP_{x,1}$)
ma = max($FA_{x,4}, DP_{x,4}$)

$$A = \begin{pmatrix} mi-1 & FA_{x,1} & FA_{x,2} & FA_{x,3} & FA_{x,4} & ma+1 \\ 0 & FA_{y,1} & FA_{y,2} & FA_{y,3} & FA_{y,4} & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} mi-1 & DP_{x,1} & DP_{x,2} & DP_{x,3} & DP_{x,4} & ma+1 \\ 0 & DP_{y,1} & DP_{y,2} & DP_{y,3} & DP_{y,4} & 0 \end{pmatrix}$$

```
Recherche des points d'intersection
for i ≤ 5 do
  for j ≤ 5 do
    if A_i et B_j pas en double then
      a = [A_i A_i]
      b = [B_j B_{j+1}]
      if a et b joints then
        if a et b pas verticales then
          if a et b parallèles then
            Ajout d'un point d'intersection (soit A_i, soit B_j)
          else if a et b quelconques then
            Calcul du point d'intersection
            Ajout du point d'intersection
          end if
        else if a et b parallèles et verticales then
          Ajout d'un point d'intersection (soit A_i, soit B_j)
        else if a pas verticale et b verticale then
          Calcul du point d'intersection
          Ajout du point d'intersection
        else if a verticale et b pas verticale then
          Calcul du point d'intersection
          Ajout du point d'intersection
        end if
      end if
    end if
  end for
end for
Trouver le segment minimum
if Pas de point d'intersection then
  Segment minimum : FA ou DP
else
  Trouver les coordonnées de FA sur les abscisses de DP
  Trouver les coordonnées de DP sur les abscisses de FA
  Trouver le segment minimum
end if
Calcul de la possibilité
Calculer le maximum du segment minimum pour trouver la possibilité P
```

FIG. A.2 : Pseudo-Code de la première version de la fonction de calcul de la possibilité

En prenant le cas le moins favorable, le coût de calcul de cette première version peut être évalué à une centaine d'opérations à virgule flottante :

$$FLO_{Poss1} \approx 100FLOP$$

A.2.2 Pseudo-code de la deuxième version de la fonction de calcul de la possibilité

Pour pouvoir être exécuté de manière efficace sur GPU, on doit éviter de faire autant de tests pour calculer la possibilité. Dans l'optique d'une parallélisation future sur GPU de l'algorithme, nous avons donc réfléchi à une autre manière de calculer la possibilité à partir des fonctions d'appartenance et densité de possibilité. Le principe est de faire une interpolation sur les segments de la fonction d'appartenance et la densité de possibilité. Les points d'interpolation doivent être identiques sur la fonction d'appartenance et la densité de possibilité. Le pas d'interpolation est calculé de manière à avoir obtenir une certaine précision sur la valeur de la possibilité (par exemple 0.01). Le pseudo-code de cette deuxième version de la fonction de calcul de la possibilité est présenté sur la figure A.3

P = Possibility2(FA,DP)
precision = 0.01
mi = min($FA_{x,1}, DP_{x,1}$)
ma = max($FA_{x,4}, DP_{x,4}$)
Trouver le segment le plus petit
dFAx = diff(FAx)
dDPx = diff(DPx)
$m_{seg} = min(dFAx, dDPx)$
Calcul du pas sur les ordonnées puis sur les abscisses
$stepY = m_{seg} \times precision$
Calcul de $stepX$ à partir de l'equation du segment minimum et de $stepY$
Création des segments
$$A = \begin{pmatrix} mi - stepX & FA_{x,1} & FA_{x,2} & FA_{x,3} & FA_{x,4} & ma + stepX \\ 0 & FA_{y,1} & FA_{y,2} & FA_{y,3} & FA_{y,4} & 0 \end{pmatrix}$$
$$B = \begin{pmatrix} mi - stepX & DP_{x,1} & DP_{x,2} & DP_{x,3} & DP_{x,4} & ma + stepX \\ 0 & DP_{y,1} & DP_{y,2} & DP_{y,3} & DP_{y,4} & 0 \end{pmatrix}$$
Point d'interpolation
$I = [mi : stepX : ma]$
Interpolation
$IFA = interp(FAx, FAy, I)$
$IDP = interp(DPx, DPy, I)$
Calcul de la possibilité
$P = max(min(IFA, IDP))$

FIG. A.3 : Pseudo-Code de la deuxième version de la fonction de calcul de la possibilité

Pour résumer, le coût de calcul de cette deuxième version se résume aux calculs des points d'interpolations I. Le coût de calcul va donc dépendre du nombre de points d'interpolation que l'on va noter N_I qui dépend de la forme des fonctions d'appartenance et des densités de possibi-

lité. Le calcul des points d'interpolations est réalisé à partir de l'équation de la droite entre 2 points A et B :

$$y = \frac{y_B - y_A}{x_B - x_A}(x - x_A) + y_A$$

Cela se résume donc à calculer le coefficient directeur $\frac{y_B - y_A}{x_B - x_A}$, ce qui correspond à 2 soustractions et 1 division, puis à faire 1 soustraction entre $x - x_A$ et enfin 1 multiplication et 1 addition. Cela correspond en tout à 6 opérations à virgule flottante.

Le nombre d'opérations à virgule flottante pour cette deuxième version du calcul de la possibilité peut donc être exprimé de la manière suivante :

$$FLO_{Poss2} \approx 2 \times N_I \times 6FLOP$$

En prenant $N_I = 10000$, cela donne $FLO_{Poss2} \approx 120000$ FLOP, soit un facteur 100 par rapport à la première version de la fonction.

A.2.3 Pseudo-code de la troisième version de la fonction de calcul de la possibilité

La deuxième version de la fonction de calcul de la possibilité permet de s'affranchir des branchements mais augmente considérablement le coût de calcul. Nous avons réfléchi, dans l'optique d'une parallélisation future sur GPU, à un autre moyen de calculer la possibilité en évitant les branchements et sans augmenter le coût de calcul. On se place dans le cas de l'algorithme G-FRI, c'est-à-dire dans le cas où les fonctions d'appartenance sont rectangulaires et les densités de possibilité trapezoïdales et dans le cas où la valeur maximale de ces deux fonctions est 1.

L'idée serait de définir deux droites pour la fonction d'appartenance (droites horizontales car les fonctions d'appartenance sont rectangulaires) à partir des points définissant la fonction d'appartenance et deux droites de la même manière pour la densité de possibilité. La figure A.4 résume le principe de cette méthode sur un cas particulier.

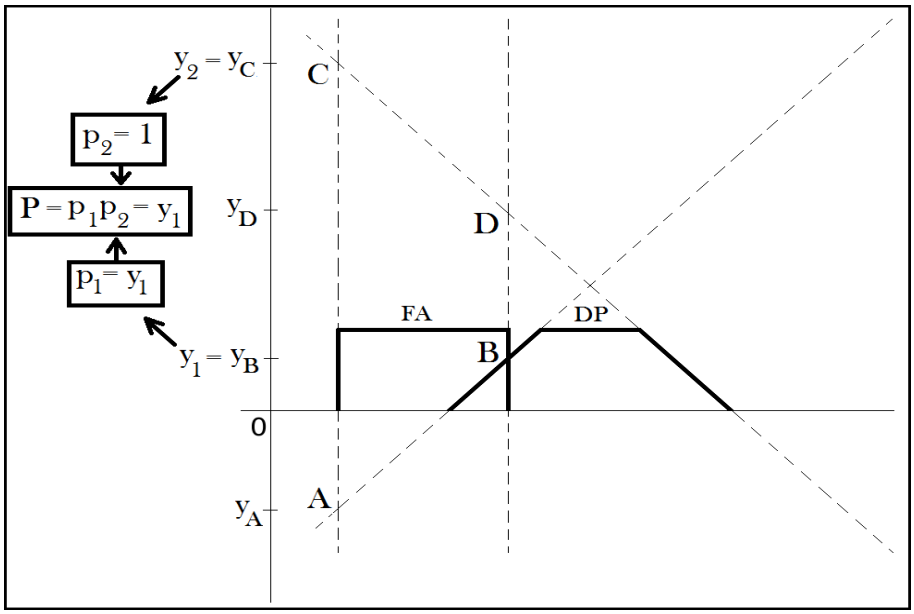


FIG. A.4 : Troisième version de la fonction de calcul de la possibilité

L'idée est de calculer les quatres points d'intersections entre ces droites (2 points (A et B) entre la première droite de la densité de possibilité et les deux droites de la fonction d'appartenance et deux points (C et D) entre la deuxième droite de la densité de possibilité et les deux droites de la fonction d'appartenance). Ensuite, on raisonne sur l'ordonnée de ces points. On définit tout d'abord $y_1 = max(y_A, y_B)$ l'ordonnée maximale des points A et B et $y_2 = max(y_C, y_D)$ celle des points C et D. Ensuite, on raisonne de la façon suivante avec le point y_1 :

- si $y_1 < 0$ alors $p_1 = 0$;
- si $0 < y_1 < 1$ alors $p_1 = y_1$;
- si $y_1 > 1$ alors $p_1 = 1$;

et de la même manière avec le point y_2 :

- si $y_2 < 0$ alors $p_2 = 0$;
- si $0 < y_2 < 1$ alors $p_2 = y_2$;
- si $y_2 > 1$ alors $p_2 = 1$;

Au final, la possibilité est tout simplement la multiplication de p_1 avec p_2 :

$$P = p_1 \; p_2$$

Cette troisième version du calcul des possibilités repose sur les hypothèses suivantes :

- La fonction d'appartenance est rectangulaire ;
- La densité de possibilité est trapézoïdale ;
- Le minimum de la densité de possibilité et de la fonction d'appartenance est égal à 0 ;
- Le maximum de la densité de possibilité et de la fonction d'appartenance est égal à 1.

Ces hypothèses sont vérifiées dans le cadre de l'algorithme G-FRI. En ajoutant quelques tests supplémentaires, cette méthode est généralisable pour toutes les formes de fonctions d'appartenance et de densités de possibilité.

Cette méthode se résume donc aux calculs des quatre points d'intersection entre les droites. Cela consiste à résoudre quatre systèmes d'équation, ce qui représente environ 5 opérations à virgule flottante par système d'équation. En tout, cette méthode engendre une vingtaine d'opérations à virgule flottante, tout en limitant le nombre de branchements. Même si cette troisième version n'a pas été implémentée durant la thèse, celle-ci semble être meilleure que les deux autres, notamment dans la perspective d'une parallélisation sur GPU. C'est la raison pour laquelle nous avons utilisé cette version pour estimer les coûts de calcul des algorithmes basés sur la logique floue (cf. 6.5).

$$FLO_{Poss3} \approx 20FLOP$$

A.3 Publications liées à la thèse

Revue nationale

- **Algorithmes de reconnaissance NCTR et parallélisation sur GPU**
  T. Boulay, N. Gac, A. Mohammad-Djafari et J. Lagoutte
  (accepté en avril 2013) Traitement du Signal, 2013

Conférences internationales

- **A Fuzzy-Logic based non cooperative target recognition**
  T. Boulay, J.Lagoutte, A. Mohammed-Djafari and N. Gac
  Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS), p. 410-415
  Naples, Italie, Novembre 2012 (présentation orale)

- **High-Dimensional Range profile geometrical visualization and performance estimation of radar target classification via a Gaussian mixture model**
  T. Boulay, A. Mohammad-Djafari, N. Gac et J. Lagoutte
  GSI2013 - Geometric Science of Information
  Paris, France, Août 2013 (poster)

Conférences nationales

- **Algorithmes de Reconnaissance Non Coopérative de Cibles et implémentation sur GPU**
  T. Boulay, A. Mohammad-Djafari, N. Gac et J. Lagoutte
  23ème colloque GRETSI sur le traitement du signal et des images
  Bordeaux, France, Septembre 2011 (présentation orale)

Présentation à des évenements scientifiques

- **Parallélisation sur GPU d'un algorithme de reconnaissance de cibles radar**
  T. Boulay, N. Gac, A. Mohammad-Djafari et J. Lagoutte
  Ecole d'été francophone de traitement d'image sur GPU
  Grenoble, France, Juin 2013 (présentation orale)

- **Implémentation et optimisation du calcul sur GPUs FERMI et non FERMI de la distance e entre des profils distance obtenus a partir de donnees radar dans le contexte NCTR**
  T. Boulay, N. Gac, A. Mohammad-Djafari et J. Lagoutte
  Ecole d'été francophone de traitement d'image sur GPU
  Grenoble, France, Juin 2011 (poster)

- **Algorithm development for NCTR function / Parallel computing on GPU**
  T. Boulay, J. Lagoutte, N. Gac et A. Mohammad-Djafari
  Computing PhdDay 2011
  Thales Recherche et Technologie (TRT), Palaiseau, Novembre 2011 (présentation orale)

- **A Fuzzy-Logic based Non Cooperative Target Recognition algorithm**
  T. Boulay, J. Lagoutte, N. Gac et A. Mohammad-Djafari
  Computing PhdDay 2012
  Thales Recherche et Technologie (TRT), Palaiseau, Novembre 2012 (poster)

Bibliographie

(2012). *NVIDIA CUDA C Programming Guide Version 5.0*. (Cité pages ix, x, 108, 109, 110, 111, 112 et 124.)

Atrouz, B., Aait Ouazzou, H., and Kimouche, H. (2007). Features influence on targets classification performance using the high range resolution profiles (HRR profiles). In *Radar Systems, 2007 IET International Conference*, pages 1 –4. (Cité page 25.)

Bandfield, J. D. and Raftery, A. E. (1993). Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics*, 49 :803–821. (Cité page 73.)

Barrientos, R. J., Gómez, J. I., Tenllado, C., Matias, M. P., and Marin, M. (2011). kNN query processing in metric spaces using GPUs. In *Proceedings of the 17th international conference on Parallel processing - Volume Part I*, Euro-Par'11, pages 380–392, Berlin, Heidelberg. Springer-Verlag. (Cité page 114.)

Belkin, M. and Nigoyi, P. (2002). Laplacian Eignemaps and spectral techniques for embedding and clustering. pages 585–591. (Cité page 78.)

Bishop, C. (2006). *Pattern recognition and machine learning*. Information science and statistics. Springer. (Cité page 22.)

Bolz, J., Farmer, I., Grinspun, E., and Schröoder, P. (2003). Sparse matrix solvers on the GPU : conjugate gradients and multigrid. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 917–924, New York, NY, USA. ACM. (Cité page 107.)

Boulay, T. (2010). Développement d'algorithmes de reconnaissance de cibles non coopératives et parallélisation sur GPU. Rapport de stage, Thales Air Sytems, Telecom ParisTech, L2S. (Cité page 45.)

Bouveyron, C. (2006). *Modélisation et classification des données de grande dimension*. PhD thesis, Université Joseph Fourier-Grenoble 1. (Cité pages 73 et 74.)

Brousseau, C. (2012). Development of multiresolution hierarchical trees to non-cooperative target recognition. In *Communications (COMM), 2012 9th International Conference on*, pages 51–54. (Cité page 37.)

Cattell, R. B. (1966). The Scree Test For The Number Of Factors. *Multivariate Behavioral Research*, 1(2) :245–276. (Cité page 76.)

Celeux, G. and Govaert, G. (1993). Gaussian parsimonious clustering models. Technical Report RR-2028, INRIA. (Cité page 73.)

Chen, B., Liu, H., and Bao, Z. (2005). PCA and kernel PCA for radar high range resolution profiles recognition. In *Proc. IEEE Int. Radar Conf*, pages 528–533. (Cité page 32.)

Chen, C., Mu, D., Zhang, H., and Hong, B. (2012). A GPU-accelerated Approximate Algorithm for Incremental Learning of Gaussian Mixture Model. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1937–1943. (Cité page 116.)

Chen, L. and Tang, H. (2004). Improved computation of beliefs based on confusion matrix for combining multiple classifiers. *Electronics Letters*, 40(4) :238 – 239. (Cité page 37.)

Chen, S. S., Donoho, D. L., and Saunders, M. A. (1999). Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1) :33–61. (Cité page 39.)

Chevalier, F. (2002). *Principles of Radar and Sonar Signal Processing*. Artech House radar library. Artech House. (Cité page 17.)

Chitchian, M., van Amesfoort, A. S., Simonetto, A., Keviczky, T., and Sips, H. J. (2013). Adapting Particle Filter Algorithms to Many-Core Architectures. In *27th IEEE International Parallel and Distributed Processing Symposium*. (Cité page 113.)

Collange, S. (2010). Analyse de l'architecture GPU Tesla. Technical report, DALI, ELIAUS, Université de Perpignan. (Cité pages 120 et 121.)

Cooke, S. J., Vlasov, A. N., Levush, B., Chernyavskiy, I. A., and Antonsen, T. M. (2011). GPU-accelerated 3D time-domain simulation of vacuum electron devices. In *Proceedings of the 2011 IEEE International Vacuum Electronics Conference*, IVEC '11, pages 305–306, Washington, DC, USA. IEEE Computer Society. (Cité page 113.)

Dessauer, M., Hitchens, J., and Dua, S. (2010). GPU-enabled high performance feature modeling for ATR applications. In *Aerospace and Electronics Conference (NAECON), Proceedings of the IEEE 2010 National*, pages 92–98. (Cité page 113.)

Donoho, D. L. and Tsaig, Y. (2008). Fast Solution of l1-Norm Minimization Problems When the Solution May Be Sparse. *IEEE Transactions on Intelligent Transportation Systems*, 54(11) :4789–4812. (Cité page 39.)

Du, L. and al. (2006). A two-distribution compounded statistical model for Radar HRRP target recognition. (Cité pages 17 et 28.)

Du, L., Liu, H., and Bao, Z. (2008). Radar HRRP Statistical Recognition : Parametric Model and Model Selection. 56(5) :1931–1944. (Cité pages 28, 36 et 71.)

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition. (Cité pages 22 et 62.)

Fuqiu, W., Zhang, W.-Q., and Jia, L. (2012). GPU accelerated GMM supervectors for speaker and language recognition. In *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, volume 1, pages 536–539. (Cité page 115.)

Garcia, V. and al. (2010). Fast GPU-based implementations and application to high-dimensional feature matching. In *IEEE Int Image Processing (ICIP) Conf*. (Cité page 114.)

Gupta, K. and Owens, J. (2009). Three-layer optimizations for fast GMM computations on GPU-like parallel processors. In *Automatic Speech Recognition Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 146–151. (Cité page 116.)

Hinton, G. E. and Roweis, S. T. (2002). Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840. (Cité page 78.)

Huang, L., Liu, Z., Yan, Z., Liu, P., and Cai, Q. (2012). An Implementation of High Performance Parallel KNN Algorithm Based on GPU. In *Networking and Distributed Computing (ICNDC), 2012 Third International Conference on*, pages 30–30. (Cité page 114.)

Hudson, S. and Psaltis, D. (1993). Correlation filters for aircraft identification from radar range profiles. *Aerospace and Electronic Systems, IEEE Transactions on*, 29(3) :741 –748. (Cité page 9.)

Jacobs, S. P. (1997). *Automatic target recognition using high-resolution radar range-profiles*. PhD thesis, St. Louis, MO, USA. AAI9733725. (Cité pages 27 et 71.)

Kantardzic, M. (2011). *Data Mining(Concepts, Models, Methods, and Algorithms)*. Wiley-Interscience. (Cité page 62.)

Kent, S., Kasapoglu, N. G., and Kartal, M. (2008). Radar target classification based on support vector machines and High Resolution Range Profiles. In *Proc. IEEE Radar Conf. RADAR '08*, pages 1–6. (Cité page 33.)

Kirk, D. and Hwu, W.-M. (2010). *Programming Massively Parallel Processors*. NVIDIA, Elsevier Inc. (Cité page 111.)

Kittler, J., Society, I. C., Hatef, M., Duin, R. P. W., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 :226–239. (Cité page 37.)

Kocsor, A. and Toth, L. (2004). Kernel-based feature extraction with a speech technology application. *Signal Processing, IEEE Transactions*, 52(8) :2250 – 2263. (Cité page 32.)

Kosir, P., DeWall, R., and Mitchell, R. (1995). A multiple measurement approach for feature alignment. In *Aerospace and Electronics Conference, 1995. NAECON 1995., Proceedings of the IEEE 1995 National*, volume 1, pages 94 –101 vol.1. (Cité page 17.)

Kotsiantis, S. B. (2007). Supervised Machine Learning : A Review of Classification Techniques. *Informatica*, 31 :249–268. (Cité page 62.)

Krüger, J. and Westermann, R. (2003). Linear Algebra Operators for GPU Implementation of Numerical Algorithms. *ACM Transactions on Graphics*, 22 :908–916. (Cité page 107.)

Kuang, Q. and Zhao, L. (2009). A Practical GPU Based KNN Algorithm. In *Proc. of the 2nd Sympsium International Computer Science and Computational Technology (ISCSCT)*, pages 151–155. (Cité page 114.)

Lee, J. A. and Verleysen, M. (2007). Nonlinear dimensionality reduction. *Springer*. (Cité page 78.)

Lefohn, A., Cates, J. E., and Whitaker, R. T. (2003). Interactive, GPU-Based Level Sets for 3D Segmentation. In *In : Medical Image Computing and Computer Assisted Intervention (MICCAI*, pages 564–572. (Cité page 107.)

Li, H.-J. and Yang, S.-H. (1993). Using range profiles as feature vectors to identify aerospace objects. *Antennas and Propagation, IEEE Transactions on*, 41(3) :261 –268. (Cité page 9.)

Li, J., Shen, L., and Yang, S. (2008). A Novel Radar Target Recognition Algorithm Based on SVM. In *Proc. Int. Symp. Intelligent Information Technology Application Workshops IITAW '08*, pages 431–434. (Cité page 33.)

Liang, S., Liu, Y., Wang, C., and Jian, L. (2010). Design and evaluation of a parallel k-nearest neighbor algorithm on CUDA-enabled GPU. In *Web Society (SWS), 2010 IEEE 2nd Symposium on*, pages 53–60. (Cité page 114.)

Liao, X. and Bao, Z. (1998). Two new categories of shift-invariant features of high-resolution radar range profiles. In *Signal Processing Proceedings, 1998. ICSP '98. 1998 Fourth International Conference on*, volume 2, pages 1485 –1488 vol.2. (Cité page 17.)

Lin, J. and Lin, J. (2010). Accelerating BP Neural Network-Based Image Compression by CPU and GPU Cooperation. In *Proc. Int Multimedia Technology (ICMT) Conf*, pages 1–4. (Cité page 113.)

Linde, Y., Buzo, A., and Gray, R. (1980). An Algorithm for Vector Quantizer Design. *Communications, IEEE Transactions on*, 28(1) :84 – 95. (Cité page 32.)

Liria, E., Higuero, D., Abella, M., de Molina, C., and Desco, M. (2012). Exploiting Parallelism in a X-ray Tomography Reconstruction Algorithm on Hybrid Multi-GPU and Multi-core Platforms. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 867–868. (Cité page 113.)

Liu, C. and Rubin, D. B. (1995). ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5 :19–39. (Cité page 71.)

Liu, H., Su, H., and Bao, Z. (2005a). Radar High Range Resolution Profiles Feature Extraction Based on Kernel PCA and Kernel ICA. In Wang, J., Liao, X., and Yi, Z., editors, *Advances in Neural Networks-ISNN 2005*, volume 3496 of *Lecture Notes in Computer Science*, pages 913–918. Springer Berlin / Heidelberg. (Cité page 33.)

Liu, H., Yang, Z., He, K., and Bao, Z. (2005b). Radar high range resolution profiles recognition based on wavelet packet and subband fusion. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP '05)*, volume 5, pages 445–448. (Cité page 37.)

Liu, X., Gao, M., and Fu, X. (2006). A Nearest Neighbor Fuzzy Classifier for Radar Target Recognition Using Combined Features. In *Signal Processing, 2006 8th International Conference on*, volume 3. (Cité page 25.)

Liu, X., Wang, B., Xu, X., Liang, J., Ren, J., and Wei, C. (2011). Modified nearest neighbor fuzzy classification algorithm for ship target recognition. In *Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference on*, pages 2254 –2258. (Cité page 25.)

Machlica, L., Vanek, J., and Zajic, Z. (2011). Fast Estimation of Gaussian Mixture Model Parameters on GPU Using CUDA. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*, pages 167–172. (Cité page 115.)

Mallat, S. G. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12) :3397–3415. (Cité page 39.)

Martin, J. and Mulgrew, B. (1990). Analysis of the theoretical radar return signal form aircraft propeller blades. In *Radar Conference, 1990., Record of the IEEE 1990 International*, pages 569–572. (Cité page 7.)

Martin, J. and Mulgrew, B. (1992). Analysis of the effects of blade pitch on the radar return signal from rotating aircraft blades. In *Radar 92. International Conference*, pages 446–449. (Cité page 7.)

Mclachlan, G. and Peel, D. (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1 edition. (Cité page 28.)

Mengdao, X. and Zheng, B. (2001). The properties of range profile of aircraft. In *Proc. Radar CIE Int. Conf*, pages 1050–1054. (Cité page 15.)

Mingjing, L., Zhefeng, Z., and Ming, H. (2009). Radar target recognition based on combined features of high range resolution profiles. In *Synthetic Aperture Radar, 2009. APSAR 2009. 2nd Asian-Pacific Conference on*, pages 876 –879. (Cité page 24.)

Moruzzis, M. and Colin, N. (1998). Radar target recognition by Fuzzy Logic. *IEEE Aerospace and Electronic Systems Magazine*, 13(7) :13–20. (Cité page 34.)

Moruzzis, M., Ferrier, J., Gosselin, F., Enert, P., and L.Lupinski (2004). MILORD : Synthesis and perspectives. In *RADAR 2004-International Conference on Radar Systems*. (Cité pages 7 et 35.)

Nabiyouni, M. and Aghamirzaie, D. (2012). A Highly Parallel Multi-class Pattern Classification on GPU. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 148–155. (Cité page 113.)

Ngo, L. T., Mai, D. S., and Nguyen, M. U. (2012). GPU-based acceleration of interval type-2 fuzzy c-means clustering for satellite imagery land-cover classification. In *Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on*, pages 992–997. (Cité page 117.)

Nuo, Z. and al. (2010). High Range Resolution Profile Automatic Target Recognition Using Sparse Representation. *Chinese Journal of Aeronautics*, pages 556 – 562. (Cité page 39.)

NVIDIA (2007). White Paper : Accelerating MATLAB with CUDA Using MEX Files. (Cité page 117.)

Owens and al (2007). A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26 :80–113. (Cité page 113.)

Parzen, E. (1962). *On estimation of a probability density function and mode*, volume 33. Annals of mathematical statistics. (Cité page 22.)

Piazza, E. (1999). Radar signals analysis and modellization in the presence of JEM application to civilian ATC radars. *Aerospace and Electronic Systems Magazine, IEEE*, 14(1) :35–40. (Cité page 7.)

Rasmussen, C. E. (2000). The Infinite Gaussian Mixture Model. In *In Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press. (Cité page 143.)

Redner, R. and Walker, H. (1984). Mixture Densities, Maximum Likelihood and the EM algorithm. *SIAM Review*, 26(2) :195–239. (Cité page 29.)

Rihaczek, A. and Herschowitz, S. (2000). *Theory and Practice of Radar Target Identification*. (Cité page 14.)

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, pages 2323–2326. (Cité page 78.)

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. In *IEEE Transactions on Computer*, pages 401–409. (Cité page 78.)

Sawall, S., Ritschl, L., Knaup, M., and Kachelriess, M. (2011). Performance comparison of openCL and CUDA by benchmarking an optimized perspective backprojection. In *3rd Workshop on High Performance Image Reconstruction*, pages 15–8. (Cité page 129.)

Shi, L., Wang, P., Liu, H., Xu, L., and Bao, Z. (2011). Radar HRRP Statistical Recognition With Local Factor Analysis by Automatic Bayesian Ying-Yang Harmony Learning. 59(2) :610–617. (Cité pages 28, 29 et 36.)

Skolnik, M. I. (2002). *Introduction to Radar systems*. McGraw-Hill. (Cité page 28.)

Srikanthan, S., Krishnan, V., and Kumar, A. (2011). Online accelarated implementation of the Fuzzy C-means algorithm with the use of the GPU platform. In *Computer and Communication Technology (ICCCT), 2011 2nd International Conference on*, pages 385–388. (Cité page 116.)

Tait, P. (2005). *Introduction to Radar Target Recognition*. IEE Radar, Sonar and Navigation series 18. The institution of Electrical Engineers. (Cité page 7.)

Tao, Y., Lin, H., and Bao, H. (2010). GPU-Based Shooting and Bouncing Ray Method for Fast RCS Prediction. *Antennas and Propagation, IEEE Transactions on*, 58(2) :494–502. (Cité page 113.)

Tenenbaum, J. B., Silva, V., and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500) :2319–2323. (Cité page 78.)

Tong, C., Huang, Z., Yan, P., Geng, F., and Guang-cai, C. (2005). Studies of modulation mechanism of jet engine modulation effect. In *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings*, volume 3, pages 4 pp.–. (Cité page 7.)

van der Heiden, R. (1998). *Aircraft Recognition with Radar Range Profile*. PhD thesis, University of Amsterdam, ftp://ftp.wins.uva.nl/pub/computer-systems/aut-sys/reports/ThesisRvdH.pdf. (Cité page 9.)

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9 :2579–2605. (Cité page 78.)

van Dorp, P., Otten, M., and Verzeilberg, J. (2012). Coherent Multistatic ISAR imaging. In *Radar Systems (Radar 2012), IET International Conference on*, pages 1–6. (Cité page 7.)

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer, 2 edition. (Cité page 31.)

Volkov, V. (2010). Better performance at lower occupancy. In *GPU Technology Conference*. (Cité pages ix, 120, 121 et 123.)

Wang, C. and Xie, J. (2011). Radar high resolution range profile target recognition based on T-mixture model. In *Proc. IEEE Radar Conf. (RADAR)*, pages 762–767. (Cité page 28.)

Wang, P., Dai, F., Pan, M., Du, L., and Liu, H. (2011). Radar HRRP target recognition in frequency domain based on autoregressive model. In *Radar Conference (RADAR), 2011 IEEE*, pages 714 –717. (Cité page 36.)

Webb, A. (2002). *Statistical pattern recognition*. Wiley. (Cité pages 22 et 27.)

Webb, A. R. (2000). Gamma mixture models for target recognition. *Pattern Recognition*, pages 2045 – 2054. (Cité page 27.)

Wehner, D. R. and Barnes, B. (1994). *High-Resolution Radar*. Artech House Publishers. (Cité page 9.)

Weinberger, K. Q., Sha, F., and Saul, L. K. (2004). Learning a kernel matrix for non linear dimensionality reduction. In *Proceedings of the International Conference on Machine Learning*. (Cité page 78.)

Wong, S. K. (2004). Non cooperative target recognition in the frequency domain. *IEEE Radar and Sonar*. (Cité pages 17 et 35.)

Xu, F. and Mueller, K. (2007). Real-time 3D computed tomographic reconstruction using commodity graphics hardware. *Physics in Medicine and Biology*, 52(12) :3405–3419. (Cité page 113.)

Xu, L. (1998). Rival penalized competitive learning, finite mixture, and multisets clustering. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 2525 –2530 vol.3. (Cité page 28.)

Yang, B., Lu, K., Liu, J., Wang, X., and Gong, C. (2012). GPU accelerated Monte Carlo simulation of deep penetration neutron transport. In *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*, pages 899–904. (Cité page 113.)

Yudanov, D., Shaaban, M., Melton, R., and Reznik, L. (2010). GPU-based simulation of spiking neural networks with real-time performance & high accuracy. In *Proc. Int Neural Networks (IJCNN) Joint Conf*, pages 1–8. (Cité page 113.)

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8 :338–353. (Cité page 33.)

Zhang, N., shan Chen, Y., and Wang, J.-L. (2010). Image parallel processing based on GPU. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 3, pages 367–370. (Cité page 113.)

Zoric, D. P., Olcan, D. I., and Kolundzija, B. M. (2013). GPU accelerated computation of radar cross sections with multiple excitations. In *Antennas and Propagation (EuCAP), 2013 7th European Conference on*, pages 2656–2659. (Cité page 113.)

Zwart, J. P., van der Heiden, R., Gelsema, S., and Groen, F. (2003). Fast translation invariant classification of HRR range profiles in a zero phase representation. *IEE Proceedings -Radar, Sonar and Navigation*, 150(6) :411–418. (Cité pages 17 et 18.)

Zyweck, A. and Bogner, R. E. (1996). Radar target classification of commercial aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, 32 :598–606. (Cité pages 9, 17 et 36.)

Titre Développement d'algorithmes pour la fonction NCTR
Application des calculs parallèles sur les processeurs GPU

Résumé Le thème principal de cette thèse est l'étude d'algorithmes de reconnaissance de cibles non coopératives (NCTR). Il s'agit de faire de la reconnaissance au sein de la classe "chasseur" en utilisant le profil distance. Nous proposons l'étude de quatres algorithmes : un basé sur l'algorithme des KPPV, un sur les méthodes probabilistes et deux sur la logique floue. Une contrainte majeure des algorithmes NCTR est le contrôle du taux d'erreur tout en maximisant le taux de succès. Nous avons pu montrer que les deux premiers algorithmes ne permettait pas de respecter cette contrainte. Nous avons en revanche proposé deux algorithmes basés sur la logique floue qui permettent de respecter cette contrainte. Ceci se fait au détriment du taux de succès (notamment sur les données réelles) pour le premier des deux algorithmes. Cependant la deuxième version de l'algorithme a permis d'augmenter considérablement le taux de succès tout en gardant le contrôle du taux d'erreur. Le principe de cet algorithme est de caractériser, case distance par case distance, l'appartenance à une classe en introduisant notamment des données acquises en chambre sourde. Nous avons également proposé une procédure permettant d'adapter les données acquises en chambre sourde pour une classe donnée à d'autres classes de cibles. La deuxième contrainte forte des algorithmes NCTR est la contrainte du temps réel. Une étude poussée d'une parallélisation de l'algorithme basé sur les KPPV a été réalisée en début de thèse. Cette étude a permis de faire ressortir les points à prendre en compte lors d'une parallélisation sur GPU d'algorithmes NCTR. Les conclusions tirées de cette étude permettront par la suite de paralléliser de manière efficace sur GPU les futurs algorithmes NCTR et notamment ceux proposés dans le cadre de cette thèse.

Mots-clés NCTR, CLASSIFICATION, RADAR, HRD, GPU, KPPV, LOGIQUE-FLOUE, PROFIL DISTANCE, RECONNAISSANCE

Title Algorithm developpment for NCTR function
Parallel computing application on GPU cards

Abstract The main subject of this thesis is the study of algorithms for non-cooperative targets recognition (NCTR). The purpose is to make recognition within "fighter" class using range profile. The study of four algorithms is proposed : one based on the KNN algorithm, one on probabilistic methods and two on fuzzy logic. A major constraint of NCTR algorithms is to control the error rate while maximizing the success rate. We have shown that the two first algorithms are not sufficient to fulfill this requirement. On the other hand, two algorithms based on fuzzy logic have been proposed and meet this requirement. Compliance with this condition is made at the expense of success rate (in particular on real data) for the first of the two algorithms based on fuzzy-logic. However, a second version of the algorithm has greatly increased the success rate while keeping

control of the error rate. The principle of this algorithm is to make classification range bin by range bin, with the introduction of data acquired in an anechoic chamber. We also proposed a procedure for adapting the data acquired in an anechoic chamber for a class to another class of targets. The second major constraint algorithms NCTR is the real time constraint. An advanced study of a parallelization on GPU of the algorithm based on KNN was conducted at the beginning of the thesis. This study has helped to identify key points of a parallelization on GPU of NCTR algorithms. Findings from this study will be used to parallelize efficiently on GPU future NCTR algorithms, including those proposed in the thesis.

Keywords NCTR, CLASSIFICATION, RADAR, HRR, GPU, KNN, FUZZY-LOGIC, RANGE PROFIL, RECOGNITION