Système de vision pour la sécurité des personnes sur les remontées mécaniques

Julien Muzeau

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : SIGNAL IMAGE PAROLE TELECOMS
Arrêté ministériel : 25 mai 2016

Présentée par
Julien MUZEAU

Thèse dirigée par **Pascal BERTOLINO** et
codirigée par **Patricia LADRET**

préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Système de vision pour la sécurité des personnes sur les remontées mécaniques

Vision system for people security on skilifts

Thèse soutenue publiquement le **14 décembre 2020**,
devant le jury composé de :

Madame Michèle ROMBAUT
Professeur, Université Grenoble Alpes, Présidente
Monsieur Christophe CUDEL
Professeur, Université de Haute-Alsace, Rapporteur
Monsieur Patrick LAMBERT
Professeur émérite, Université Savoie Mont-Blanc, Rapporteur
Monsieur Christophe DUCOTTET
Professeur, Université de Saint-Etienne, Examinateur
Monsieur Fakhri KARRAY
Professeur, Université de Waterloo, Examinateur
Monsieur Pascal BERTOLINO
Maître de conférence, Université Grenoble Alpes, Directeur de thèse

Système de vision pour la sécurité des personnes sur les remontées mécaniques

Julien Muzeau

Résumé

Devant l'augmentation de la fréquentation des domaines skiables et la multiplication des accidents sur les remontées mécaniques imputés au comportement humain, la sécurité est devenue un enjeu majeur des gérants de stations.

Pour lutter contre ce phénomène, la start-up grenobloise BLUECIME a développé un système de vision par ordinateur, baptisé SIVAO, capable de détecter une situation dangereuse lors de l'embarquement d'une remontée mécanique. Le fonctionnement du système se décompose en trois étapes. D'abord, le siège (ou véhicule) est détecté dans l'image. Par la suite, la présence de skieurs sur ce dernier est confirmée ou infirmée. Enfin, la position du garde-corps est déterminée. Si des passagers sont présents sur le véhicule et si le garde-corps n'est pas abaissé, alors la situation est considérée comme dangereuse. Dans ce cas, une alarme est déclenchée afin d'alerter les skieurs ou l'opérateur qui peut alors ralentir le télésiège pour sécuriser le véhicule.

Malgré des résultats convaincants, de nombreuses difficultés s'opposent à SIVAO : variabilités diverses (taille du véhicule, orientation de l'embarquement, conditions météorologiques, nombre de passagers), vibrations de la caméra, configuration complexe dans le cadre d'une nouvelle installation, *etc*.

Le projet MIVAO, en partenariat avec le laboratoire HUBERT CURIEN, l'entreprise BLUECIME et le groupe SOFIVAL, a donc vu le jour dans le but de pallier les difficultés précédentes. L'objectif est de construire une intelligence artificielle capable de détecter, voire d'anticiper, une situation dangereuse à bord de véhicules d'un télésiège, dans le but d'assurer la sécurité des passagers. Au sein de ce projet, l'objectif général du GIPSA-LAB est l'annotation automatique, de la manière la moins supervisée possible, de vidéos de rémontées mécaniques.

Premièrement, nous présentons une méthode de classification visant à confirmer ou infirmer la présence de passagers sur chaque véhicule. Cette information préliminaire est en effet cruciale dans l'analyse d'un danger potentiel. La technique proposée repose sur des caractéristiques fabriquées à la main et interprétables physiquement. Nous montrons qu'en incluant des connaissances *a priori*, les résultats obtenus

concurrencent ceux issus de réseaux de neurones complexes, tout en permettant un fonctionnement en temps-réel.

Ensuite, nous détaillons un processus pour le comptage des passagers de chaque véhicule de la manière la plus non-supervisée possible. Ce processus consiste en une première étape de réduction de dimension, suivie d'une procédure de partitionnement de données. Cette dernière vise dans le cadre de notre projet à regrouper les passages dont les véhicules transportent le même nombre de passagers. Par la suite, nous pouvons déduire, à partir d'un nombre réduit d'étiquettes fournies manuellement, le nombre de personnes présentes lors de chaque passage. On détaille notamment deux algorithmes développés durant cette thèse. Le premier algorithme propose une généralisation de la méthode de clustering par densité DBSCAN via l'introduction du concept de voisinage ellipsoïdal. Le deuxième concilie les clusterings par mélange Gaussien et spectral dans le but de découvrir des groupes de données non-convexes.

Dans un dernier temps, nous abordons le problème de l'extraction automatique des véhicules dans les images issues de la caméra, ainsi que de la modélisation de leur trajectoire. Pour ce faire, nous proposons une première méthode qui consiste à supprimer le bruit du flux optique grâce à l'utilisation de la déformation optique. On présente également une technique permettant de déterminer automatiquement la durée d'un passage de véhicule via une analyse fréquentielle.

De plus, nous détaillons un effort d'annotations, travail visant à détourer, au niveau pixel, les passagers et véhicules de séquences de quarante images consécutives.

Abstract

With the increase in the number of visitors in mountain ranges and the multiplication of accidents on skilifts attributed to human behavior, safety has become a major issue for resort managers.

To fight this phenomenon, the start-up from Grenoble BLUECIME developed a computer vision system, named SIVAO, which is able to detect a hazardous situation at the boarding of a skilift. The operation of the system breaks down into three steps. First, the chair (or vehicle) is detected in the image. Then, the presence of passengers is confirmed or invalidated. Finally, the position of the security railing is determined. If passengers are present on the vehicle and if the security railing is not down, then the situation is considered as hazardous. In that case, an alarm is triggered, in order to inform the skiers or the operator who can slow down the skilift to secure the vehicle.

Despite convincing results, numerous difficulties have to be overcome by SIVAO : various variabilities (vehicle size, boarding orientation, meteorological conditions, number of passengers), camera vibration, complex configuration in the context of a new plant, *etc*.

The MIVAO project, in partnership with the HUBERT CURIEN laboratory, the BLUECIME start-up and the SOFIVAL company, was born in order to overcome the previous challenges. The goal is to build an artificial intelligence able to detect, even anticipate, a hazardous situation on vehicles of a skilift, in order to guarantee the security of passengers. Within this project, the general goal of the GIPSA-LAB is the automatic annotation, in the least supervised way possible, of chairlift videos.

Firstly, we present a classification method whose aim is to confirm or invalidate the presence of passengers on each vehicle. In fact, this preliminary information is critical for the analysis of a potential danger. The proposed technique is based on hand-made features which have a physical interpretation. We show that, by including *a priori* knowledge, the obtained results are comptetitive against those from complex neural networks, allowing real-time functioning as well.

Then, we detail a process for passenger counting on each vehicle in the most unsupervised way possible. This pipeline consists in a dimensionality reduction step

followed by a data clustering stage. The latter aims, in the context of our project, at gathering tracks whose vehicles carry the same number of passengers. One can then deduce, from a small number of labels obtained by hand, the number of people present during each track. In particular, we detail two algorithms developed during this thesis. The first one proposes a generalisation of the density-based clustering method DBSCAN, via the introduction of the concept of ellipsoidal neighborhood. The second conciliates Gaussian mixture and spectral clusterings so as to discover non-convex data groups.

Finally, we address the problem of automatic extraction of vehicles from camera images, as well as the modeling of their trajectory. To do this, we propose a first method which consists in removing the noise from the optical flow by means of the optical strain. We also present a technique for automatically determining the duration of a vehicle track via frequency analysis.

Moreover, we detail an annotation work whose objective is to define clipping paths, pixel by pixel, over the passengers and vehicles in sequences of fourty consecutive images.

Remerciements

Un doctorat n'est pas un long fleuve tranquille, mais c'est un voyage qui mérite tout de même d'être réalisé. A travers nombre de péripéties, j'ai pu apprendre énormément, du point de vue scientifique mais également du côté humain. Je vais essayer dans la suite de remercier chaque personne qui m'a permis d'arriver à destination.

Je voudrais commencer par remercier mes encadrants, Patricia et Pascal, pour ces trois belles années. D'abord pour leur patience et leur persévérance dans l'obtention du financement de ce projet. Ensuite pour l'autonomie qu'ils m'ont donnée et leurs conseils scientifiques particulièrement affûtés. Egalement pour les différentes opportunités d'enseignement. Je pourrais encore énumérer une dizaine de raisons de les remercier, toutes aussi importantes les unes que les autres. Pour résumer, merci à eux de m'avoir fait découvrir et de m'avoir transmis leur passion pour le métier d'enseignant-chercheur.

Par la même, je remercie Maria pour les deux années de collaboration. J'ai apprécié ces longues discussions remplies d'idées, d'astuces, de conseils. J'ai beaucoup appris, notamment sur la manière de rédiger un article scientifique. J'espère que ces échanges t'ont également été bénéfiques.

Je souhaiterais ensuite vivement remercier les membres de mon jury. Merci à Christophe CUDEL et Patrick LAMBERT pour la lecture attentive de mon manuscrit et leurs remarques/questions pertinentes. Merci à Christophe DUCOTTET et Fakhri KARRAY pour leur présence par vidéo lors de ma soutenance et leur intérêt pour les travaux présentés. Merci enfin à Michèle ROMBAUT d'avoir accepter la présidence du jury. Malheureusement, la soutenance n'a pas pu se tenir en présentiel du fait de la pandémie de Covid-19, j'espère toutefois pouvoir tous vous (re)voir prochainement.

Je voudrais adresser mes remerciements aux différents partenaires du consortium : la région Auvergne-Rhône-Alpes pour le financement, Bluecime pour avoir porter le projet, le laboratoire Hubert Curien et le groupe Sofival pour leurs remarques et questions intéressantes. Tout particulièrement, merci à Matthieu pour toutes ces

discussions, scientifiques et/ou personnelles, dans un bureau ou au pied d'un mur d'escalade.

Le laboratoire est presque devenu une deuxième maison pendant trois ans, merci donc à toutes les personnes qui s'y sont trouvées : Bruce pour m'avoir (quasiment) remis en condition physique, Ludo (dit le phasme) pour les échanges sur des thèmes divers et variés, surtout musicaux, Ivan pour le savoir informatique et pas que, Thibaut, Ali, Tien, Jitu, Saloua, Karina, Dora, Dawood, Pedro, Hélène. Merci également à Gipsadoc pour tous les efforts en ce qui concerne la vie sociale du Gipsa-lab.

Les amis de longue date ont également beaucoup compté pendant mon doctorat, même s'il est vrai que les réponses aux messages se sont fait attendre de mon côté, clairement trop. Merci à Antoine, Michaël, Archis, Nico, Johanna, Romain...

Merci également à toute ma famille pour le soutien inconditionnel toutes ces années. Maintenant que c'est terminé, je n'aurais plus à répondre à la question : "Peux-tu m'expliquer SIMPLEMENT ce que tu fais ?". Une mention spéciale pour mes parents qui n'ont jamais douté.

Pour finir, je souhaiterais remercier celle qui m'a soutenu et surtout supporté pendant ces trois ans. Tu as su, entres autres, me remotiver quand ça n'allait pas et vanter la qualité esthétique de mes articles. Merci Anaïs pour tout.

Table des matières

Table des sigles et acronymes

ACP	Analyse en Composantes Principales
ADCN	Anisotropic Density-based Clustering with Noise
ADL	Analyse Discriminante Linéaire
ADQ	Analyse Discriminante Quadratique
AE	Auto-Encodeur
AIC	Akaike Information Criterion
AV	Addition de Vecteurs
BIC	Bayesian Information Criterion
CC	Composantes Connexes
4C	Computing Clusters of Correlation Connected Objects
CD	Cartes de Diffusion
CEM	Classification Expectation Maximization
CLINK	Complete-LINKage clustering
CLIQUE	CLustering In QUEst
CS	Clustering Spectral
DBSCAN	Density-Based Spatial Clustering with Applications to Noise
DIANA	DIvisive ANAlysis clustering
DO	Déformation Optique
DOM	Détection d'Objets Mobiles
EC-LGOF	Entropy-based Canny operator with Local and Global Optical Flow
EM	Espérance-Maximisation
EN-DBSCAN	Ellipsoidal Neighborhood Density-Based Spatial Clustering with Applications to Noise
FCM	Fuzzy C-Means
FN	Faux Négatifs
FO	Flux Optique
FP	Faux Positifs

GDBSCAN	Generalized Density-Based Spatial Clustering with Applications to Noise
GSC	Gaussian Spectral Clustering
HDBSCAN	Hierarchical Density-Based Spatial Clustering with Applications to Noise
IMED	IMage Euclidean Distance
JSON	JavaScript Object Notation
LLE	Locally-Linear Embedding
LTSA	Local Tangent Space Alignment
MDS	Multi-Dimensional Scaling
MG	Mélange Gaussien
MIVAO	Montagne Innovante, Vision et Apprentissage par Ordinateur
MMG	Modèle de Mélange Gaussien
MML	Minimum Message Length
OPTICS	Ordering Points To Identify the Clustering Structure
PAM	Partitioning Around Medoids
PID	Proportionnel-Intégral-Dérivé
PPC	Prédictions Positives Correctes
RL	Régression Logistique
RM	Remontée Mécanique
SDEQ	Somme des Distances Euclidiennes Quadratiques
SEM	Stochastic Expectation Maximization
SIFT	Scale-Invariant Feature Transform
SIVAO	Système Intelligent de Vision Artificielle par Ordinateur
SLINK	Single-LINKage clustering
SOFT	Statistical Optical Flow Thresholding
STING	STatistical INformation Grid
STRMTG	Service Technique des Remontées Mécaniques et des Transports Guidés
SVM	Support-Vector Machines
t-SNE	t-distributed Stochastic Neighbor Embedding
UCB	Upper Confidence Bounds

UMAP	Uniform Manifold Approximation and Projection
UPGMA	Unweighted Pair Group Method with Arithmetic mean
VN	Vrais Négatifs
VP	Vrais Positifs
WFM	Weighted Fowlkes-Mallows

Nomenclature

Dans la suite de ce document, nous utilisons les règles suivantes. Un scalaire, c'est-à-dire un nombre réel ou complexe, est désigné par une lettre minuscule (*e.g.* n ou γ). Un caractère minuscule en gras, par exemple $\boldsymbol{n}$ ou $\boldsymbol{\gamma}$, représente un vecteur à une seule colonne. Enfin, une matrice est affichée via une lettre majuscule en gras, *e.g.* $\boldsymbol{N}$ ou $\boldsymbol{\Gamma}$.

De plus, précisons que le symbole « □ », rencontré en fin de démonstration, signifie Ce Qu'il Fallait Démontrer (CQFD) et représente la fin de cette dernière.

1 Introduction

Sommaire

1.1 Contexte général

Le tourisme en milieu montagnard s'est particulièrement développé ces vingt dernières années. C'est notamment le cas de la France qui pointe aujourd'hui à la deuxième position internationale, juste derrière les Etats-Unis, pour ce qui est du nombre de journées-skieurs, c'est-à-dire de visites journalières d'une personne pratiquant un sport de glisse sur un domaine skiable. En effet, parmi les cinquante plus grandes stations dans le monde, 80% se trouvent dans les Alpes européennes. [1]

Cet essor peut s'expliquer de plusieurs manières :

— accessibilité des domaines skiables rendue aisée.

— prix des forfaits attractifs et communication via les outils numériques.

— modernisation des remontées mécaniques réduisant l'attente.

— utilisation de la neige de culture pour allonger la saison hivernale et ouverture en été pour pratiquer, entre autres, le vélo de montagne.

Quoi qu'il en soit, cette augmentation du nombre de personnes présentes sur les pistes engendre une multiplication des dangers potentiels et donc une recrudescence du nombre d'accidents intervenant chaque année : c'est particulièrement vrai sur les Remontées Mécaniques (RM). En effet, une étude [2] menée par le Service Technique des Remontées Mécaniques et des Transports Guidés (STRMTG) a analysé 108 accidents graves intervenus entre 2006 et 2014. Ce dernier a pu noter que 74% de ces accidents se déroulent à l'embarquement ou au débarquement. De plus, les statistiques révèlent que le comportement humain en est à l'origine dans 92% des cas. On peut citer par exemple l'oubli de la barrière de sécurité, la surcharge d'un siège, la perte d'un ski... Il est donc dans l'intérêt des gérants de domaines skiables d'anticiper et d'empêcher ce genre d'accidents.

1.2 Projet SIVAO

A Grenoble, Isère (38), une société a décidé de se pencher sur le sujet. BLUECIME, fondée en 2015, développe alors un système de vision par ordinateur capable de détecter une situation dangereuse lors de l'embarquement : ce dernier est baptisé Système Intelligent de Vision Artificielle par Ordinateur (SIVAO).

1. 2019 International Report on Snow & Mountain Tourism – Laurent Vanat
2. Domaines Skiables de France n°39, juillet 2015, pages 28 et suivantes

Introduisons d'abord les premiers termes du vocabulaire-métier. Un siège accroché au câble principal est appelé **véhicule**. On identifie la partie haute de ce dernier par **joint**. Enfin, la rambarde qui permet de retenir les passagers à l'intérieur du véhicule est dénommée **garde-corps**. Une illustration est donnée en Figure 1.1b.

(a) Image issue de la caméra et zone de détection en vert (résolution = 1280 × 1024).

(b) Image unitaire et véhicule. Le joint est représenté en rouge, le garde-corps en bleu (résolution = 237 × 237).

Figure 1.1. Illustration des types d'images et du vocabulaire-métier sur une remontée mécanique.

Revenons à présent à SIVAO. L'embarquement d'une remontée mécanique est observé grâce à une caméra fixée sur le premier pylône du télésiège. Le fonctionnement du système peut ensuite être décomposé en trois étapes :

1. D'abord, le véhicule est détecté dans l'image.
2. Par la suite, la présence de skieurs sur ce dernier est confirmée ou infirmée.
3. Enfin, la position du garde-corps (haute ou basse) est déterminée.

Apportons quelques précisions. L'étape 1 consiste à localiser le véhicule dans la **zone de détection**, parallélépipède projeté en deux dimensions sur les images issues de la caméra. Cette zone, représentée en vert sur la Figure 1.1a, délimite la région de l'espace dans laquelle le siège est cherché. Une fois celui-ci trouvé, des vignettes (appelées **images unitaires**) centrées sur le véhicule sont extraites : un exemple de telles images est donné en Figure 1.1b. On désigne par **passage** l'ensemble des positions d'un véhicule à l'intérieur de la zone de détection. De plus, à l'étape 3, la position du garde-corps est dite *basse* lorsque celle-ci est complètement abaissée (comme dans la Figure 1.1b), *haute* lorsqu'elle est totalement levée. Pour conclure, si des passagers sont présents sur le véhicule et si le garde-corps n'est pas en position basse, alors la situation est considérée comme dangereuse. Dans ce cas, une alarme visuelle et sonore est déclenchée, afin d'alerter les skieurs ou l'opérateur qui peut alors ralentir ou arrêter temporairement le télésiège pour sécuriser le véhicule.

A l'heure actuelle, le système SIVAO permet une valeur de rappel et une mesure de spécificité égales à 93,5% et 98,8% respectivement vis-à-vis de la détection d'une

situation dangereuse. Il est à noter que ces valeurs sont acquises par des tests sur une base de données de plus de 60 000 passages de véhicules enregistrés sur trois stations et vingt-et-une remontées mécaniques.

1.3 Difficultés rencontrées

Malgré les résultats convaincants présentés précédemment, de nombreuses difficultées s'opposent à SIVAO et doivent être surmontées.

La première est liée au travail en milieu extérieur. En effet, les images à traiter présentent de ce fait des variabilités : certaines sont observables en Figure 1.2, où trois images de quatre remontées mécaniques sont présentées. Ces variabilités peuvent être catégorisées en :

— **inter-RM**, c'est-à-dire celles qui varient d'une remontée mécanique à l'autre, telles que la taille du véhicule, l'orientation de l'embarquement, *etc*.

— **intra-RM**, c'est-à-dire celles qui affectent une même rémontée mécanique comme les conditions météorologiques, l'illumination, le nombre de passagers, la saison, *etc*.

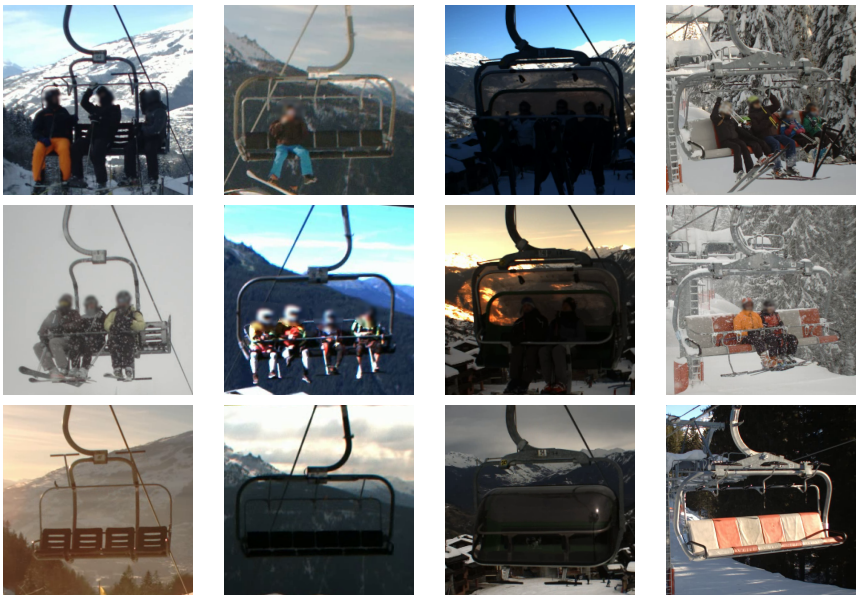


Figure 1.2. Variabilités liées au projet SIVAO représentées par trois images issues de quatre remontées mécaniques. Les visages sont floutés pour des raisons de confidentialité.

Un deuxième défi provient des vibrations de la caméra. En effet, cette dernière étant fixée à un pylône de la remontée mécanique, les véhicules à proximité engendrent des vibrations au niveau des pylônes et rendent donc complexe la détection et le suivi des sièges à tout instant.

Enfin, le dernier point se rapporte à la configuration du système SIVAO dans le cadre d'une nouvelle installation. Les différentes éléments mentionnés dans la section précédente sont fixés manuellement : zone de détection, masque du joint, seuils algorithmiques, boîtes englobantes, *etc*. Ainsi, un déploiement de cette solution en l'état actuel à grande échelle, voire même à l'international, semble inenvisageable car chronophage et coûteux.

1.4 Projet MIVAO

Le projet Montagne Innovante, Vision et Apprentissage par Ordinateur (MIVAO) a donc vu le jour dans le but de pallier les difficultés précédentes. Il est le fruit d'un consortium entre :

— la start-up BLUECIME (Grenoble, 38).

— le laboratoire GIPSA-LAB (Grenoble, 38) spécialisé en automatique, signal et image, parole et cognition.

— le laboratoire HUBERT CURIEN (Saint-Etienne, 42) expert en optique, informatique et télécommunication.

— le groupe SOFIVAL gérant les stations du Val d'Isère.

L'objectif du projet MIVAO est, à terme, de construire une intelligence artificielle capable de détecter, voire d'anticiper, une situation dangereuse à bord de véhicules d'un télésiège, et ce dans le but d'assurer la sécurité des clients de domaines skiables. Evidemment, cette solution doit être insensible aux variabilités citées en section 1.3, *i.e.* à la météo, à la remontée, au type de véhicule...

D'un point de vue pratique, les rôles au sein du projet sont répartis comme suit : SOFIVAL traite de l'infrastructure informatique, les deux laboratoires proposent des solutions aux problèmes posés (basées sur l'apprentissage profond pour HUBERT CURIEN, orientées vision par ordinateur pour GIPSA-LAB) et BLUECIME implémente les algorithmes développés dans le but d'obtenir un fonctionnement optimisé en temps-réel.

Les efforts associés à MIVAO se concentrent sur vingt-et-une remontées mécaniques implantées dans trois stations. Toutefois, celles-ci sont anonymisées dans l'intégralité de ce document pour des raisons de confidentialité demandée par BLUECIME.

1.5 Objectifs du partenaire et de la thèse

Dans le cadre de ce projet, l'objectif général du GIPSA-LAB est l'annotation automatique, de la manière la moins supervisée possible, de vidéos de rémontées mécaniques. En outre, les tâches associées sont :

- la détection des véhicules et la modélisation de leur trajectoire.
- la création du masque de détection.
- la détermination de la zone de détection optimale.
- la confirmation d'une présence humaine sur chaque véhicule.
- le comptage non-supervisé des passagers de chaque siège.

Ces objectifs pratiques seront développés et atteints en tant qu'applications des différents apports de ce travail de thèse.

Les trois premiers points permettent de caractériser le système étudié d'un point de vue physique et de faciliter et d'accélérer la configuration lors de l'installation d'un nouveau télésiège. En ce qui concerne les deux derniers items, ils traitent du comptage de passagers, qui est important à plusieurs égards. C'est en effet une information préliminaire cruciale dans l'analyse du danger. De plus, il permet d'estimer en temps-réel le flux de skieurs sur l'ensemble du domaine. Enfin, en cas de panne prolongée, le nombre de passagers encore présents sur la remontée mécanique conditionne les moyens à mettre en oeuvre pour les évacuer.

Précisons ici que les données auxquelles nous avons accès sont des vidéos, issues de la caméra, d'une durée d'environ deux minutes, et régulièrement échantillonnées tout au long de l'année.

1.6 Organisation du manuscrit

La suite de ce document est décomposée en trois parties et sept chapitres.

La Partie I (chapitre 2) présente une méthode de classification binaire vide/non-vide, c'est-à-dire visant à confirmer ou infirmer la présence de passagers sur chaque

véhicule, indépendamment de leur nombre exact. Cette information préliminaire est en effet cruciale dans l'analyse d'un danger potentiel. La méthode proposée repose sur des caractéristiques fabriquées à la main et interprétables physiquement. Nous montrons qu'en incluant des connaissances *a priori*, les résultats obtenus concurrencent ceux issus de réseaux de neurones complexes, tout en permettant un fonctionnement en temps-réel. Cela illustre le principe de simplicté (ou rasoir d'Occam) dans le sens où "les hypothèses suffisantes les plus simples doivent être préférées". Ce travail a donné lieu à une publication :

> J. Muzeau, P. Ladret et P. Bertolino, "Linear classification of chair-lift images for presence analysis", International Conference on Quality Control by Artificial Vision, 2019. [Muz+19]

En Partie II (chapitres 3, 4, 5, 6) est détaillé un processus pour le comptage des passagers de chaque véhicule de la manière la plus non-supervisée possible. Ce processus, expliqué dans le chapitre 3, consiste en une première étape de réduction de dimension, suivie d'une procédure de partitionnement de données, puis d'une dernière étape de transduction. Le chapitre 4 offre un état de l'art des techniques de ce domaine et détaille en particulier celle choisie, à savoir les Cartes de Diffusion (CD) [Tal+13]. Cette première étape est indispensable car elle permet d'extraire de l'information utile à partir de séquences d'images en grande résolution. Le chapitre 5 procède de la même manière vis-à-vis du partitionnement de données (ou *clustering*), qui vise dans le cadre de notre projet à regrouper les passages dont les véhicules transportent le même nombre de passagers. On présente notamment deux algorithmes développés durant cette thèse. Le premier algorithme propose une généralisation de la fameuse méthode de clustering par densité DBSCAN [Est+97]. La densité locale, approchée classiquement par des sphères, est dans ce cas déterminée grâce à l'introduction du concept de **voisinage ellipsoïdal**, ce qui permet une adaptation automatique à n'importe quel jeu de données. Ce travail a été soumis pour révision à une conférence internationale et est en attente d'acceptation :

> J. Muzeau, M. Oliver-Parera, P. Ladret et P. Bertolino, "EN-DBSCAN : Density-Based Clustering through Adaptive Ellipsoidal Neighborhood", Winter Conference on Applications of Computer Vision, 2021 (en attente d'acceptation).

Le deuxième concilie les clusterings par mélange Gaussien et spectral dans le but de partitionner des données complexes, pas uniquement celles présentant une distribution sphérique. Ce travail a été valorisé par une publication :

> J. Muzeau, M. Oliver-Parera, P. Ladret et P. Bertolino, “Combining Mixture Models and Spectral Clustering for Data Partitioning”, International Conference on Image Analysis and Recognition, 2020. [Muz+20]

Finalement, le chapitre 6 présente les principaux résultats obtenus par le processus complet appliqué aux images de télésièges.

Quant à la Partie III (chapitre 7), elle aborde le problème de l’extraction automatique des véhicules dans les images issues de la caméra, ainsi que de la modélisation de leur trajectoire. Pour ce faire, nous proposons une méthode qui consiste à supprimer le bruit du Flux Optique (FO) et ce grâce à l’utilisation de la Déformation Optique (DO). Ce travail a donné lieu à une publication :

> M. Oliver-Parera, J. Muzeau, P. Ladret et P. Bertolino, “Contour Detection of Multiple Moving Objects in Unconstrained Scenes using Optical Strain”, International Conference on Digital Image Computing : Techniques and Applications, 2020 (en attente de publication).

Précisons que, contrairement aux deux parties précédentes qui opèrent directement sur les images unitaires, la Partie III vise à fournir des annotations de vidéos, telles que la durée d’un passage, la localisation spatiale des véhicules, *etc*.

Enfin, le chapitre 8 présente la conclusion des travaux effectués durant cette thèse et une discussion sur les directions futures à suivre.

Ajoutons également que l’Appendice D détaille un effort d’annotations, travail visant à détourer, au niveau pixel, les véhicules ainsi que les passagers dans des séquences de quarante images consécutives.

Première partie

Détection de la présence de passagers

2 Classification vide/non-vide

Sommaire

2.1 Motivations

Nous nous intéressons dans ce chapitre à la détection de présence humaine sur chacun des véhicules. Cette information est précieuse car tout traitement ultérieur devient inutile si aucun passager n'est présent sur le siège, la sécurité est en effet totalement assurée dans ces cas-là. Dans la suite, nous qualifierons un véhicule de **vide** si aucun skieur n'y est assis, **non-vide** dans le cas où au moins un passager est présent.

La solution apportée jusqu'à présent par BLUECIME à ce problème se base sur des boîtes englobantes situées au niveau des torses ou jambes des passagers supposés. En comptant le nombre de pixels qui sont en mouvement, c'est-à-dire qui n'appartiennent pas à l'arrière-plan, et en le comparant à un seuil prédéfini, on obtient à chaque instant l'information de présence de skieurs sur chaque siège. Elle peut de plus être rafinée en prenant en compte son évolution temporelle à travers la zone de détection. On aperçoit immédiatement l'inconvénient de ce procédé : la nécessité de définir des paramètres supplémentaires (boîtes englobantes, choix du travail sur les torses ou jambes, seuil de détection), souvent complexes et chronophages à fixer.

Nous proposons donc dans ce chapitre une approche d'apprentissage automatique (*machine learning*) qui permet de s'abstraire des difficultés mentionnées précédemment. Elle est basée sur une analyse discriminante appliquée à des caractéristiques physiques interprétables. En incluant des informations *a priori* sur notre système, la méthode proposée s'avère être facile d'utilisation et autorise un fonctionnement en temps-réel, nécessaire pour notre application. Particulièrement simple, notre approche donne cependant de meilleures performances que d'autres techniques de classification plus complexes et rivalisent même avec des algorithmes à base de réseaux de neurones.

2.2 Analyse discriminante

Notre choix concernant la technique de classification se porte sur l'**analyse discriminante**. Cette dernière figure effectivemnt parmi les plus simples à mettre en oeuvre et à exécuter, dû principalement à son faible nombre de paramètres. La suite de cette section détaille l'Analyse Discriminante Linéaire (ADL) ainsi que sa variante plus complexe, l'Analyse Discriminante Quadratique (ADQ).

2.2.1 Analyse discriminante linéaire

L'ADL, dans sa forme actuelle, a été mise au point par Ronald Fisher en 1936 alors qu'il travaillait sur la catégorisation d'espèces d'iris [Fis36b]. C'est avant tout une technique de classification linéaire, d'où son nom, mais elle peut également permettre une réduction de dimension supervisée (qui n'est pas abordée dans ce chapitre). De plus, on peut l'appréhender soit d'un point de vue "géométrique", soit d'un point de vue probabiliste.

Le but de l'ADL, en tant qu'outil de classification, est de **déterminer les hyperplans qui permettent de discriminer au mieux différentes classes**.

Considérons le cas d'un jeu de n données $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ où chacun des $\boldsymbol{x_i}$ est un vecteur de caractéristiques (ou *features*) en d dimensions. A chaque donnée est associé un scalaire y_i représentant la classe à laquelle $\boldsymbol{x_i}$ appartient. Par souçi de simplicité, nous considérons ici le cas de deux classes $\mathcal{C}_0$ et $\mathcal{C}_1$, ainsi $y_i \in \{0, 1\}$. n_0 données appartiennent à la classe $\mathcal{C}_0$, n_1 à $\mathcal{C}_1$ et donc $n_0 + n_1 = n$. Evidemment, ce qui suit peut être généralisé au cas de classes multiples. Introduisons de plus les moyennes et matrices de covariance de chaque classe j :

$$\boldsymbol{\mu_j} = \frac{1}{n_j} \sum_{\boldsymbol{x_i} \in \mathcal{C}_j} \boldsymbol{x_i}, \tag{2.1}$$

$$\boldsymbol{\Sigma_j} = \frac{1}{n_j - 1} \sum_{\boldsymbol{x_i} \in \mathcal{C}_j} (\boldsymbol{x_i} - \boldsymbol{\mu_j})(\boldsymbol{x_i} - \boldsymbol{\mu_j})^T. \tag{2.2}$$

La méthode d'ADL consiste donc à déterminer l'hyperplan, paramétrisé par son vecteur normal $\boldsymbol{w} \in \mathbb{R}^d$ et son ordonnée à l'origine w_o, de telle manière que la projection des données sur celui-ci amène à une séparation optimale des différentes classes. La règle de décision, afin de déterminer la classe d'une nouvelle donnée $\boldsymbol{x}$, peut ainsi s'écrire

$$\boldsymbol{w}^T \boldsymbol{x} \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} w_o, \tag{2.3}$$

ce qui signifie que la classe $\mathcal{C}_0$ (respectivement $\mathcal{C}_1$) est choisie pour $\boldsymbol{x}$ si le terme de gauche est supérieur (respectivement inférieur) à w_o.

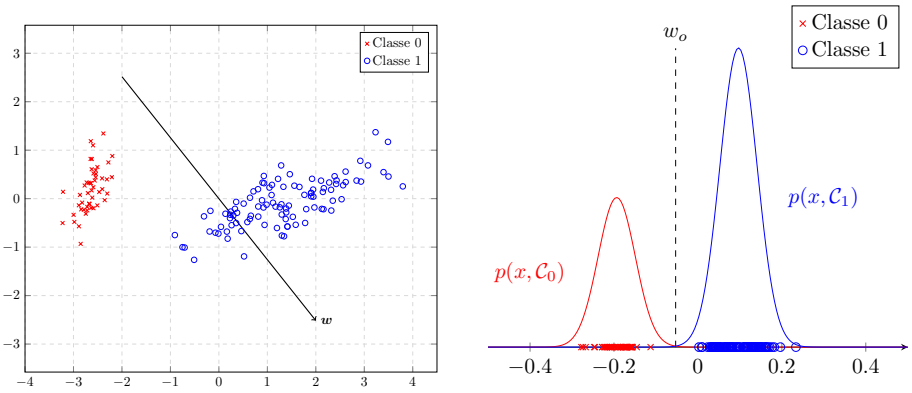
En pratique, on peut calculer le vecteur normal grâce à la relation

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_0} - \boldsymbol{\mu_1}), \tag{2.4}$$

où $\boldsymbol{\Sigma}$ est la moyenne pondérée des matrices de covariance de chaque classe (Équation 2.2). Autrement dit :

$$\boldsymbol{\Sigma} = \frac{n_0 - 1}{n_0 + n_1 - 2}\boldsymbol{\Sigma_0} + \frac{n_1 - 1}{n_0 + n_1 - 2}\boldsymbol{\Sigma_1}. \tag{2.5}$$

Arrêtons nous un instant et analysons les résultats obtenus sur un exemple simple. Le jeu de données, inspiré de la base de données sur les espèces d'iris [Fis36a], est présenté en Figure 2.1. Il est composé de deux classes $\mathcal{C}_0$ ($n_0 = 50$) et $\mathcal{C}_1$ ($n_1 = 100$). Le vecteur normal $\boldsymbol{w}$ est représenté par la flèche noire, à gauche. On peut dans un premier temps noter que ce vecteur ne joint pas directement les moyennes des deux classes, il présente une légère rotation de telle manière à prendre en compte la distribution des deux nuages de point : ce principe, proposé par Fisher, est lié à la matrice de covariance de l'Équation 2.4. Une fois $\boldsymbol{w}$ déterminé, les données originales sont projetées sur cet axe et la Figure 2.1b est obtenue. Le scalaire w_o est ensuite calculé comme étant l'intersection entre les densités de probabilités jointes de chaque classe.



(a) Jeu de données. **(b)** Probabilités jointes.

Figure 2.1. ADL appliquée à un exemple bidimensionnel à deux classes.

Précisons que l'**hypothèse de distribution normale** est utilisée ici, que ce soit pour les données originales ou celles projetées sur l'hyperplan de séparation. De surcroît, on suppose que les variances et matrices de covariance sont égales, c'est-à-dire que les différentes classes sont distribuées de la même manière : on parle alors d'**hypothèse d'homoscédasticité**. Cette dernière est en réalité implicite dans l'Équation 2.4 : seule $\boldsymbol{\Sigma}$, moyenne pondérée des matrices de covariance de chaque classe, y apparaît.

Enfin, il est possible d'obtenir une expression explicite pour l'ordonnée à l'origine sous la forme

$$w_o = \frac{\mu_0 + \mu_1}{2} + \frac{\sigma^2}{\mu_1 - \mu_0} \ln \frac{n_0}{n_1}, \tag{2.6}$$

où μ_0, μ_1, σ sont les moyennes et écart-type des deux classes projetées. Des compléments sont donnés en Appendice A.

2.2.2 Analyse discriminante quadratique

L'analyse discriminante peut également être étudiée d'un point de vue probabiliste, en comparant les densités *a posteriori* dans l'espace original (en dimension d). Pour une nouvelle donnée $\boldsymbol{x}$, la règle de décision devient :

$$p(\mathcal{C}_0|\boldsymbol{x}) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} p(\mathcal{C}_1|\boldsymbol{x}) \tag{2.7}$$

$$p(\boldsymbol{x}|\mathcal{C}_0)p(\mathcal{C}_0) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) \tag{2.8}$$

$$\frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma_0}|^{1/2}} \exp\left(-\frac{1}{2}d_0(\boldsymbol{x})\right) \cdot \frac{n_0}{n} \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma_1}|^{1/2}} \exp\left(-\frac{1}{2}d_1(\boldsymbol{x})\right) \cdot \frac{n_1}{n} \tag{2.9}$$

$$-\frac{1}{2}\ln|\boldsymbol{\Sigma_0}| - \frac{1}{2}d_0(\boldsymbol{x}) + \ln n_0 \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} -\frac{1}{2}\ln|\boldsymbol{\Sigma_1}| - \frac{1}{2}d_1(\boldsymbol{x}) + \ln n_1 \tag{2.10}$$

où

$$d_j(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu_j})^T \boldsymbol{\Sigma_j}^{-1} (\boldsymbol{x} - \boldsymbol{\mu_j}). \tag{2.11}$$

L'hypothèse Gaussienne est une fois de plus utilisée ici. Le problème posé à l'Équation 2.10 est résoluble : on parle alors d'ADQ [Has+09].

Trois différences majeures entre ADL et ADQ sont à souligner :

- Premièrement, $\boldsymbol{x}$ intervient de manière quadratique ici (Équation 2.11), linéaire en ADL : la résolution de ce problème est donc plus complexe, même si tout à fait faisable.
- En conséquence du point précédent, la frontière de séparation entre chaque classe est linéaire en ADL, ce qui entraîne une adaptation aux données limitée, contrairement à l'ADQ.
- Enfin, l'hypothèse d'homoscédasticité n'est pas utilisée en ADQ. Cette contrainte en moins permet ainsi à cette méthode d'être efficace dans un plus grand nombre de situations.

Les calculs sont poursuivis en Appendice A et montrent que les deux visions de l'analyse discriminante, géométrique et probabiliste, coïncident.

2.3 Features extraites

Après avoir détaillé la méthode de classification utilisée, la question naturelle qui vient est celle du choix des variables sur lequelles l'analyse discriminante sera appliquée : on parle aussi de **caractéristiques** (ou *features* en anglais). C'est un élément clé de toute technique d'apprentissage automatique. Cette sélection conditionne les étapes postérieures et les résultats finaux. Il est possible en pratique d'utiliser l'ensemble des pixels de chaque image : bien que logique, ce choix est souvent plus approprié pour des algorithmes complexes, à base de réseaux de neurones artificiels par exemple. Une autre possibilité consiste à extraire certaines valeurs d'intérêt de chaque image et de les utiliser par la suite comme points de départ pour la classification. L'intérêt majeur de cette solution est le fait que ces features possèdent une signification physique et nous permettront une compréhension et une interprétation *a posteriori* plus approfondies.

En l'occurence, nous souhaitons dans ce travail combiner, pour chaque image, des **indicateurs statistiques** et des **métriques empruntées au domaine du traitement des images**. Les features utilisées, au nombre de onze, sont données en Tableau 2.1 et sont catégorisées de deux manières : par rapport à leur provenance d'une part (*e.g.* statistiques ou traitement de l'image), par rapport au niveau de l'image auquel elles interviennent (pixel ou structurel) d'autre part. Précisons que ces caractéristiques sont calculées sur les images unitaires (telles que celles de la Figure 1.2) converties en niveaux de gris : en effet, dans le système visuel humain, l'information de couleur n'aide pas particulièrement à la compréhension de l'image, l'essentiel du contenu d'une scène est déterminé grâce aux niveaux de gris, d'où notre choix.

Détaillons les features choisies et leur intérêt pour notre application. Les six premières permettent de décrire l'image de manière statistique, jusqu'au quatrième ordre inclus. Plus précisément, considérons une image $\boldsymbol{I} = (I_{ij})$ de hauteur h et de largeur l. Ainsi :

$$\text{Moyenne } \mu = \frac{1}{hl} \sum_{i=1}^{h} \sum_{j=1}^{l} I_{ij} \tag{2.12}$$

$$\text{Maximum } max = \max_{i=1\ldots h,\, j=1\ldots l} I_{ij} \tag{2.13}$$

$$\text{Minimum } min = \min_{i=1\ldots h,\, j=1\ldots l} I_{ij} \tag{2.14}$$

Caractéristique	Type		Niveau	
	Statistique	Trait. de l'image	Pixel	Structurel
Moyenne (1)	×		×	
Maximum (2)	×		×	
Minimum (3)	×		×	
Ecart-type (4)	×		×	
Skewness (5)	×		×	
Kurtosis (6)	×		×	
Contraste (7)		×	×	
Niveau de flou (8)		×	×	
Nombre de CC (9)		×		×
Longueur des contours (10)		×		×
Variance des contours (11)		×		×

Table 2.1. Caractéristiques étudiées et leur catégorisation.

$$\text{Ecart-type } \sigma = \sqrt{\frac{1}{hl}\sum_{i=1}^{h}\sum_{j=1}^{l}(I_{ij}-\mu)^2} \tag{2.15}$$

$$\text{Skewness } sk = \frac{1}{hl}\sum_{i=1}^{h}\sum_{j=1}^{l}\left(\frac{I_{ij}-\mu}{\sigma}\right)^3 \tag{2.16}$$

$$\text{Kurtosis } kt = \frac{1}{hl}\sum_{i=1}^{h}\sum_{j=1}^{l}\left(\frac{I_{ij}-\mu}{\sigma}\right)^4 \tag{2.17}$$

Le contraste, proposé par Michelson [Mic27] et déterminé par

$$c = \frac{max - min}{max + min}, \tag{2.18}$$

de même que le niveau de flou [CR+07], nous donnent une information sur la qualité de l'image analysée. En effet, de mauvaises conditions météorologiques ou un flou de mouvement important dégradent fortement ces métriques. Quant aux trois dernières features, elles nous informent de la structure elle-même de l'image et sont modifiées par la présence de passagers. C'est le cas du nombre de Composantes Connexes (CC) et du nombre de contours, que l'on peut quantifier via la somme et la variance de leur longueur.

2.4 Résultats

Nous présentons dans cette section les premiers résultats obtenus par l'analyse discriminante appliquée aux features extraites des images unitaires en niveaux de gris.

Rappelons que les remontées mécaniques sont anonymisées, nous nous référons ainsi à chacune d'entre elles par une lettre (A, B, C...). Le Tableau 2.2 détaille la base de données complète : on peut y lire le nombre d'images (n_images) ainsi que le pourcentage d'images représentant des véhicules vides (%_vides). Les premières difficultés liées à notre application apparaîssent : l'étendue des nombres d'images est grande (de moins de 10 000 à presque 140 000) et le pourcentage vide/non-vide diffère fortement de 50% dans certains cas. Pour ces raisons, on peut qualifier cette base de données de **déséquilibrée**.

RM	n_images	%_vides	RM	n_images	%_vides
A	87143	61,7	K	29957	30,3
B	11161	27,2	L	9862	72,5
C	14330	52,9	M	97572	58,2
D	22315	58,1	N	93357	43,5
E	36493	48,1	O	26669	85,1
F	33803	14,2	P	100492	86,4
G	139058	44,1	Q	17399	58,7
H	102605	56,8	R	67771	62,9
I	20718	51,0	S	25013	76,5
J	39596	58,9			

Table 2.2. Détails sur la base de données analysée.

Du fait des variabilités inter-RM, c'est-à-dire celles qui varient d'une remontée mécanique à l'autre, la méthode détaillée précédemment est appliquée sur chaque télésiège, indépendamment des autres. Pour chaque remontée, les images associées sont divisées aléatoirement en deux sous-ensembles : l'entraînement est effectué sur le premier, soit environ deux tiers des images, puis le modèle résultant est testé sur le tier restant. Il est important de noter que la répartition des deux classes (dans la base de données complète de chaque remontée) est conservée par les sous-ensembles. Par exemple, si, pour la RM E, 40% des données correspondent à la classe vide et 60% à

la classe non-vide, alors ces pourcentages sont également respectés pour l'ensemble d'entraînement ainsi que pour celui de test. De cette manière, on garantit que le modèle appris est non-biaisé et représentatif de tous les types de situation.

Afin d'évaluer la qualité de notre méthode de classification binaire, nous utilisons les nombres de Vrais Positifs (VP), Vrais Négatifs (VN), Faux Positifs (FP) et Faux Négatifs (FN). Ces derniers sont définis en Tableau 2.3. Par exemple, FP donne le nombre d'images classifiées comme non-vides alors qu'elles sont vides en réalité. Notons enfin que la vérité-terrain a été obtenue par annotation manuelle par BLUECIME.

	Vérité-terrain	
	Vide (0)	Non-vide (1)
Vide (0)	VN	FN
Non-vide (1)	FP	VP
Prédiction		

Table 2.3. Définition des nombres VP, VN, FP et FN par rapport à la classification prédite et la vérité-terrain.

La métrique naturelle à utiliser est la **précision** (*accuracy* en anglais) :

$$\text{Précision} = \frac{VP + VN}{VP + VN + FP + FN}. \tag{2.19}$$

Toutefois, cette mesure est biaisée dans le cas de données déséquilibrées, *i.e.* quand les différentes classes ne sont pas uniformément réparties. On préfère alors employer le **F-score** [Chi92] (ou **F-mesure**), défini comme la moyenne harmonique de la valeur de **Rappel** et des **Prédictions Positives Correctes** (PPC). Plus précisément :

$$\text{Rappel} = \frac{VP}{VP + FN} \tag{2.20}$$

$$\text{PPC} = \frac{VP}{VP + FP} \tag{2.21}$$

$$\text{F-score} = 2 \cdot \left(\frac{1}{\text{PPC}} + \frac{1}{\text{Rappel}} \right)^{-1} = 2 \cdot \frac{\text{PPC} \times \text{Rappel}}{\text{PPC} + \text{Rappel}} \tag{2.22}$$

Les résultats produits par l'ADL sont donnés en Tableau 2.4. Pour chaque remontée, le processus entraînement–test est répété 30 fois afin d'assurer la stabilité des

résultats, la moyenne et l'écart-type sont affichés dans le tableau. Rappelons qu'à chaque répétition les ensembles d'entraînement et de test sont tirés aléatoirement, sans remise, parmi les images d'une remontée, tout en respectant la répartition 2/3–1/3, ainsi que la distribution de chaque classe à l'intérieur des sous-ensembles. La dernière ligne du tableau donne la moyenne de chaque métrique sur l'ensemble de la base de données. Afin de faciliter la visualisation, on représente en Figure 2.2 les F-mesures ordonnées sous la forme d'un diagramme en barres. On peut voir grâce à ce dernier graphe que les résultats sont déjà convaincants. En effet, le taux de classification moyen sur l'ensemble des 19 remontées mécaniques est égal à **85,4%**. De plus, huit remontées présentent un F-score supérieur à 91%. Enfin, le score le plus faible est 0,744, atteint pour le télésiège J, le plus élevé pour la remontée L avec 0,970.

RM	PPC	Rappel	Précision	F-mesure
A	0,740 ± 0,002	0,833 ± 0,003	0,716 ± 0,002	0,783 ± 0,002
B	0,849 ± 0,011	0,829 ± 0,012	0,914 ± 0,004	0,839 ± 0,008
C	0,899 ± 0,005	0,937 ± 0,005	0,911 ± 0,003	0,918 ± 0,003
D	0,772 ± 0,004	0,876 ± 0,005	0,778 ± 0,004	0,821 ± 0,003
E	0,897 ± 0,003	0,979 ± 0,002	0,936 ± 0,002	0,936 ± 0,002
F	0,895 ± 0,006	0,926 ± 0,005	0,974 ± 0,001	0,910 ± 0,004
G	0,830 ± 0,002	0,828 ± 0,002	0,850 ± 0,001	0,829 ± 0,002
H	0,803 ± 0,003	0,828 ± 0,003	0,787 ± 0,002	0,815 ± 0,002
I	0,747 ± 0,004	0,771 ± 0,007	0,750 ± 0,004	0,759 ± 0,004
J	0,681 ± 0,002	0,820 ± 0,004	0,668 ± 0,003	0,744 ± 0,003
K	0,809 ± 0,006	0,736 ± 0,007	0,867 ± 0,003	0,770 ± 0,005
L	0,947 ± 0,004	0,993 ± 0,002	0,955 ± 0,003	0,970 ± 0,002
M	0,754 ± 0,002	0,815 ± 0,003	0,738 ± 0,002	0,784 ± 0,002
N	0,806 ± 0,003	0,838 ± 0,004	0,842 ± 0,002	0,822 ± 0,002
O	0,859 ± 0,001	0,974 ± 0,003	0,842 ± 0,002	0,913 ± 0,001
P	0,879 ± 0,000	0,987 ± 0,001	0,871 ± 0,001	0,930 ± 0,000
Q	0,945 ± 0,003	0,983 ± 0,002	0,957 ± 0,002	0,964 ± 0,001
R	0,728 ± 0,002	0,827 ± 0,004	0,697 ± 0,002	0,774 ± 0,002
S	0,932 ± 0,002	0,947 ± 0,002	0,906 ± 0,002	0,939 ± 0,002
Moyenne	0,830	0,880	0,840	0,854

Table 2.4. Résultats de l'analyse discriminante linéaire appliquée aux images unitaires capturées sur 19 remontées mécaniques.

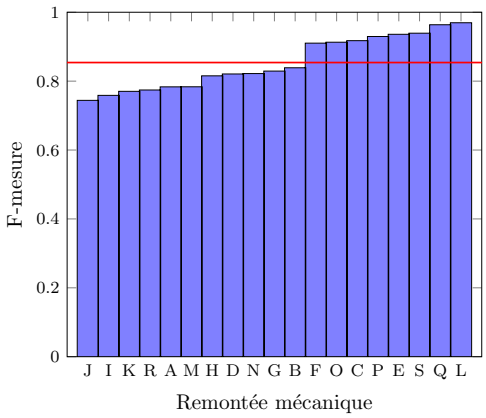


Figure 2.2. Diagramme en barres du taux de classification (F-mesure) par ADL pour chaque remontée mécanique. La moyenne est représentée par la ligne horizontale rouge.

Bien que ces premiers résultats soient satisfaisants, on peut remarquer que l'hypothèse d'homoscédasticité n'est pas vérifiée en pratique. Considérons en effet les matrices de covariance $\mathbf{\Sigma_0}$ et $\mathbf{\Sigma_1}$ de chaque classe et déterminons la matrice

$$\ln\left(1 + |\mathbf{\Sigma_0}/\mathbf{\Sigma_1}|\right), \tag{2.23}$$

où les différentes opérations sont réalisées élément par élément. Idéalement, si les matrices de covariances étaient égales, on obtiendrait une matrice dont tous les éléments sont $\ln 2 \approx 0,693$. La Figure 2.3 montre cette matrice pour la remontée mécanique B : on peut y noter que l'égalité est loin d'être atteinte et que donc **l'hypothèse d'homoscédasticité n'est pas vérifiée**.

Pour cette raison, l'utilisation de l'analyse discriminante quadratique semble plus appropriée dans certains cas. L'idée est ici de mettre en compétition l'ADL et l'ADQ. Plus précisément, pour chaque remontée, les deux modèles sont testés et celui qui aboutit à la F-mesure la plus élevée est choisi. Les résultats sur l'ensemble des télésièges sont donnés en Tableau 2.5 : ce dernier met en comparaison l'ADL seule, l'ADL combinée avec l'ADQ comme expliqué ci-dessus, et une méthode à base de réseaux de neurones [Bas+17] (reposant sur l'architecture ResNet à 50 couches). En ce qui concerne les analyses discriminantes, les F-mesures obtenues sur l'ensemble des remontées sont représentées en Figure 2.4. Notons le **gain de 1,3%** apporté par l'ADQ sur la moyenne générale. De plus, on peut observer une translation vers le haut du diagramme en boîte ce qui montre une amélioration du taux de classification pour chaque remontée. Cependant, il est clair que la méthode proposée ne concurrence pas des techniques basées sur des réseaux de neurones,

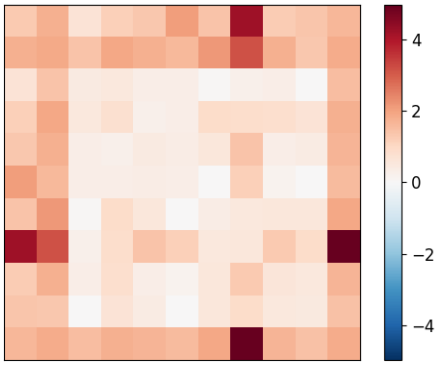


Figure 2.3. Comparaison des matrices de covariance de chaque classe (vide/non-vide) pour la remontée mécanique B. La différence est significative et l'hypothèse d'homoscédasticité est donc trop contraignante.

bien que ces résultats soient à pondérer par la complexité de ces algorithmes (plus de vingt millions de paramètres pour [Bas+17] contre douze pour notre approche) et la durée de la phase d'entraînement (quatre à cinq heures contre moins d'une heure dans le cas des analyses discriminantes).

Méthode	PPC	Rappel	Précision	F-mesure
ADL	0,830	0,880	0,840	0,854
ADL ou ADQ	0,844	0,893	0,854	0,867
[Bas+17]	0,946	0,889	0,987	0,917

Table 2.5. Classification vide/non-vide des images de remontées mécaniques par ADL seule, ADL et ADQ combinées (*i.e.* la meilleure des deux pour chaque RM), et par une méthode à base de réseaux de neurones. Résultats moyennés sur l'ensemble des télésièges.

2.5 Inclusion d'informations *a priori*

Le principal problème de la méthode proposée dans la section précédente est le fait que les différentes features sont calculées sur l'ensemble de l'image unitaire. Or, les passagers sont localisés spatialement dans cette image, sur le véhicule. On peut donc raisonnablement penser que les caractéristiques extraites sont "biaisées" par

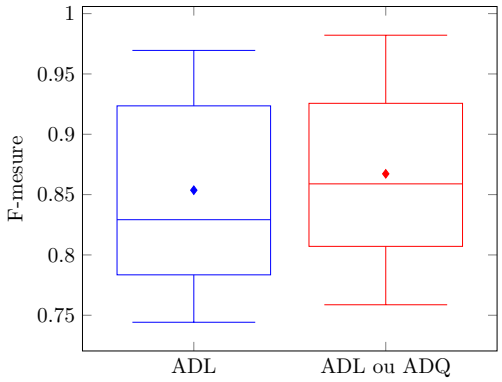


Figure 2.4. Diagrammes en boîtes des F-mesures obtenues par l'ADL et par l'ADL combinée avec l'ADQ (c'est-à-dire la meilleure des deux pour chaque télésiège) sur l'ensemble des remontées mécaniques.

des éléments non-pertinents, tels que les pixels de l'arrière-plan, et influencent la classification dans la mauvaise direction.

Afin de surmonter cette difficulté, nous proposons d'inclure dans notre processus de classification des **informations *a priori***. En l'occurence, nous choisissons d'extraire, pour chaque image unitaire, deux sous-images localisées au niveau des torses et jambes supposés des passagers. Des exemples sont donnés en Figure 2.5. Précisons que ces boîtes englobantes restent constantes pour toutes les images d'une même remontée mécanique et qu'elles ont été créées manuellement par BLUECIME.

Figure 2.5. Images unitaires séparées en deux zones : *torses* (haut, rouge) et *jambes* (bas, bleu).

Le processus de classification reste le même que jusqu'ici : les analyses discriminantes linéaire et quadratique sont appliquées sur les images unitaires complètes, les zones jambes ou les zones torses, pour un total de **six modèles différents**. Celui qui engendre le F-score le plus élevé est choisi pour chaque remontée. Les résultats obtenus sont présentés en Tableau 2.6. On y présente le F-score moyen obtenu pour

chaque télésiège ainsi que le modèle qui l'a généré. La moyenne sur la totalité des remontées mécaniques est également donnée en dernière ligne.

RM	F-mesure	Méthode		Images		
		ADL	ADQ	Unitaires	Jambes	Torses
A	0,916		×		×	
B	0,996		×			×
C	0,989	×			×	
D	0,957	×			×	
E	0,971		×		×	
F	0,981		×		×	
G	0,958	×			×	
H	0,980	×				×
I	0,939	×				×
J	0,942		×		×	
K	0,911	×			×	
L	0,998		×		×	
M	0,840		×		×	
N	0,967		×			×
O	0,960		×		×	
P	0,930	×				×
Q	0,994	×			×	
R	0,976	×			×	
S	0,994	×			×	
Moyenne	0,958					

Table 2.6. Taux de classification vide/non-vide (F-mesure) par remontée mécanique et modèle associé.

Plusieurs observations peuvent êtres faites. D'abord, on peut noter que la F-mesure moyenne est grandement améliorée (**gain d'environ dix points**) : cela montre que l'inclusion d'informations *a priori* porte ses fruits et confirme notre hypothèse de départ, qui consiste à dire que les features extraites des images unitaires sont biaisées par les pixels en arrière-plan. Cette conjecture peut également être vérifiée par le tableau en remarquant qu'aucun des modèles choisis n'utilise les images unitaires pour extraire les caractéristiques utiles à la classification. De plus, on peut observer une représentation équilibrée des analyses discriminantes linéaire et quadratique.

Cet équilibre n'est toutefois pas respecté pour ce qui est du type d'images utilisé. En effet, **les zones jambes sont préférées aux zones torses** dans 75% des cas : on peut légitimement le comprendre car les jambes, normalement au nombre de deux pour chaque passager, ainsi que les skis/planches, apportent plus de preuves de présence, notamment du point de vue des contours générés. Enfin, soulignons le fait que la méthode proposée surclasse à présent celle utilisant un réseau de neurones complexe (voir Tableau 2.5).

2.6 Analyse approfondie

Cette section vise à pousser l'analyse des résultats obtenus précédemment.

On peut d'abord se demander dans quelle mesure chaque caractéristique intervient dans l'analyse discriminante linéaire des images unitaires. Pour ce faire, étudions les coefficients des hyperplans amenant à une séparation optimale des différentes classes. Précisons que les données sont d'abord centrées puis réduites dans ce cas, ce qui permet une comparaison entre variables qui sont à une échelle semblable. La Figure 2.6 donne l'"amplitude" en valeur absolue de chaque caractéristique : plus elle est élevée, plus la feature intervient dans la classification. Deux exemples arbitraires, en pointillés, ainsi que la moyenne sur l'ensemble des remontées mécaniques sont affichés. Notons que les barres d'erreur, en rouge, représentent la moitié de l'écart-type pour chaque télésiège et que la numérotation des caractéristiques est donnée en Tableau 2.1. Nous voyons, grâce aux amplitudes moyennées, que les caractéristiques extraites n'interviennent pas au même niveau dans l'ADL. On en distingue deux prédominantes, à savoir le minimum ainsi que le contraste : cette interprétation est toutefois à prendre avec précaution étant donné les grands écarts-types associés. En revanche, deux features semblent être moins importantes : il s'agit du maximum et du nombre de composantes connexes. Enfin, la puissance de chaque feature varie grandement d'une remontée à l'autre, validant l'intuition qu'il est complexe de déterminer un modèle universel, *i.e.* unique pour n'importe quel télésiège.

Intéressons-nous maintenant au temps d'exécution de la méthode proposée. En réalité, le temps d'inférence à proprement parler est petit, car consistant en un simple produit scalaire en onze dimensions, et donc négligeable devant le temps nécessaire pour calculer les différentes caractéristiques indispensables à la classification. La Figure 2.7 présente le temps d'exécution par remontée mécanique et par type d'images utilisées. On représente en rouge la limite temps-réel fixée à 25 images/seconde, c'est-à-dire 0,04 ms. On peut d'abord remarquer que le temps d'exé-

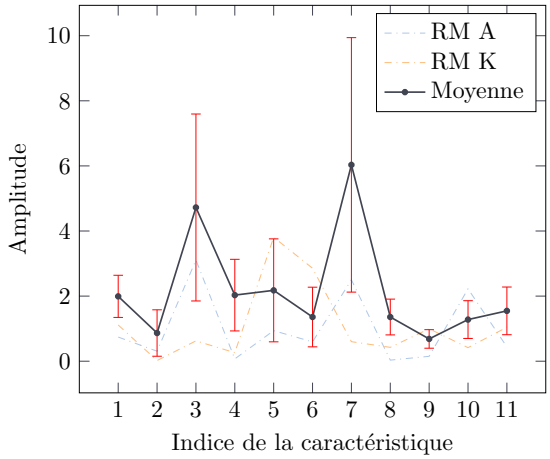


Figure 2.6. Pertinence de chaque feature dans le processus de classification vide/non-vide (ADL sur images unitaires).

cution, à type d'images fixé, n'est pas constant : ceci est lié au fait que la résolution des images varie d'une remontée à l'autre. De plus, il est compliqué d'obtenir un fonctionnement en temps-réel en utilisant les images unitaires. Toutefois, l'inclusion d'informations *a priori* (les zones torses et jambes) apporte une **division par dix du temps d'exécution**, rendant notre méthode appropriée dans des situations réelles. Soulignons attentivement que c'est un gain considérable comparé à d'autres méthodes plus complexes : en effet, l'algorithme présenté dans [Bas+17] fournit un temps d'inférence de 356 ms en moyenne. Notre méthode n'en a besoin que de 34,8 pour les images unitaires, 4,94 pour les zones torses et 4,21 pour les zones jambes, soit un rapport de 10,2, 72,1 et 84,6 respectivement.

Dans un troisième et dernier temps, attardons nous sur les difficultés rencontrées par notre méthode. En particulier, on peut voir sur le Tableau 2.6 que la remontée mécanique M est à la peine avec une F-mesure égale à 0,84, devancée de 7% par la remontée K. La Figure 2.8 donne quelques exemples du télésiège M. Ces derniers montrent à quel point il est difficile pour une méthode simple, telle que celle proposée, de s'adapter à des situations aussi complexes. D'une part, les conditions de luminosité (contre-jour et ombres principalement) rendent la tâche ardue. D'autre part, différents éléments extérieurs interfèrent : grue, vélo, poussières sur l'écran...

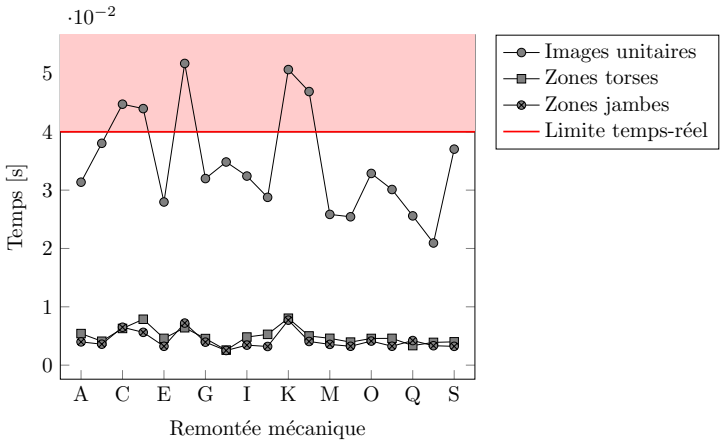


Figure 2.7. Temps d'exécution pour chaque remontée pour les images unitaires, les zones torses et les zones jambes.

Figure 2.8. Exemples de la remontée mécanique M montrant pourquoi la classification vide/non-vide y est compliquée.

2.7 Comparaison avec d'autres méthodes de classification

La dernière question qui n'a pas été abordée jusqu'ici est celle du choix de la méthode de classification. En effet, l'analyse discriminante est une des plus simples à mettre en oeuvre, si ce n'est la plus simple. Qu'en est-il de techniques légèrement plus complexes ? Nous nous attardons dans cette section sur la Régression Logistique (RL) [Cra02] et les machines à vecteurs de support [CV95], Support-Vector Machines (SVM) en anglais.

La procédure reste la même que précédemment : on ajoute toutefois une étape supplémentaire de validation croisée afin de fixer le paramètre de régularisation, présent dans les deux méthodes. Dans la suite, les résultats donnés sont calculés avec le paramètre optimal. Le Tableau 2.7 présente les F-mesures obtenues via une RL et un SVM appliqués aux images unitaires, aux zones torses et aux zone

jambes pour chaque remontée. On donne en dernière ligne la moyenne de chaque modèle sur l'ensemble des télésièges. Ces valeurs sont à mettre en comparaison avec celles du Tableau 2.6, qui sont répétées en dernière colonne. On remarque une fois encore qu'aucun modèle utilisant les images unitaires n'arrive en tête. De plus, la classification par régression logistique donne de meilleures performances en moyenne que celle par SVM. Enfin, de même que précédemment, on peut combiner les modèles utilisant les zones torses et jambes : on obtient alors une F-mesure moyenne égale à 0,928 pour la RL, 0,880 pour le SVM. Ainsi, la méthode proposée surclasse ces deux algorithmes (F-mesure moyenne égale à 0,958), en plus d'être plus simple et donc plus rapide à l'exécution.

RM	Unitaires		Jambes		Torses		Table 2.6
	RL	SVM	RL	SVM	RL	SVM	
A	0,781	0,591	0,864	0,778	0,839	0,762	**0,916**
B	0,832	0,809	0,936	0,954	**0,996**	0,990	**0,996**
C	0,878	0,606	**0,992**	0,984	0,962	0,877	0,989
D	0,776	0,542	0,915	0,685	0,782	0,556	**0,957**
E	0,925	0,806	**0,972**	0,965	0,913	0,837	0,971
F	0,800	0,796	0,964	0,926	0,947	0,969	**0,981**
G	0,568	0,536	0,793	0,686	0,779	0,721	**0,958**
H	0,746	0,560	0,969	0,934	**0,980**	0,957	**0,980**
I	0,694	0,524	0,763	0,694	0,931	0,898	**0,939**
J	0,726	0,526	0,904	0,787	0,758	0,634	**0,942**
K	0,769	0,741	0,805	0,697	0,816	0,865	**0,911**
L	0,981	0,896	**0,999**	0,982	**0,999**	0,989	0,998
M	0,771	0,614	0,790	0,692	0,734	0,589	**0,840**
N	0,809	0,576	0,847	0,833	0,933	0,935	**0,967**
O	0,921	0,719	0,923	0,841	0,935	0,851	**0,960**
P	0,927	0,864	0,927	0,863	**0,930**	0,863	**0,930**
Q	0,966	0,910	**0,998**	0,976	0,984	0,979	0,994
R	0,775	0,639	0,918	0,834	0,835	0,758	**0,976**
S	0,949	0,881	**0,994**	0,976	0,988	0,975	**0,994**
Moyenne	0,821	0,691	0,909	0,847	0,897	0,842	**0,958**

Table 2.7. Taux de classification (F-mesure) obtenu par une RL et par un SVM. Les valeurs obtenues en section 2.5 sont redonnées en dernière colonne.

2.8 Conclusion et perspectives

Nous avons proposé dans ce chapitre une méthode de classification simple permettant de séparer les images représentant des véhicules vides de ceux qui comptent des passagers, tâche essentielle pour l'analyse d'un danger potentiel. En partant de l'analyse discriminante linéaire, nous avons pu nous affranchir des contraintes de ce modèle et ajouter des informations *a priori* sur le système étudié afin d'aboutir à une classification robuste et efficace. La méthode proposée, en plus de fournir des résultats compétitifs comparés à des techniques hautement plus complexes, permet un fonctionnement en temps-réel, crucial dans notre projet. Sur une note plus philosophique, ce travail illustre le principe de parcimonie, plus connu sous le nom de rasoir d'Occam : ce dernier stipule que "les hypothèses suffisantes les plus simples doivent être préférées".

De nombreuses perspectives s'offrent à nous afin de poursuivre ce travail. Tout d'abord, il serait intéressant d'investiguer d'autres caractéristiques à extraire des images unitaires. On peut penser à des features spatiales d'une part, telles que le rapport signal à bruit, la variation totale, le contraste local ou les histogrammes de couleur. D'autre part, le contenu fréquentiel (*e.g.* via les filtres de Gabor) de chaque image peut apporter de précieuses informations. Enfin, il pourrait être bénéfique de combiner ces caractéristiques avec des outils comme les histogrammes des gradients orientés [DT05] ou des points clés classiques (Scale-Invariant Feature Transform (SIFT) [Low04] ou Harris [HS88]).

Une autre piste prometteuse concerne le choix du modèle pour chaque remontée. Nous avons pu voir en section 2.5 que tous les modèles sont représentés. On peut alors se demander si, au lieu d'utiliser celui qui amène à de meilleurs résultats en moyenne, on ne pourrait pas plutôt tous les essayer et garder celui ou ceux qui apportent le plus de confiance (incarnée, par exemple, par la distance entre la donnée considérée et la frontière de classification). Les performances au cas par cas s'en retrouveraient améliorées.

Egalement, le travail présenté incorpore des informations *a priori* sous la forme de boîtes englobantes, créées manuellement. Il pourrait être intéressant d'opter pour un processus entraînement-test afin de déterminer les zones jambes et torses optimales de manière automatique.

Finalement, nous avons vu l'hypothèse d'homoscédasticité dans l'ADL et comment elle est implémentée en pratique : via la moyenne arithmétique pondérée des matrices de covariance de chaque classe (voir Équation 2.5). Or cette moyenne

n'est pas particulièrement adaptée pour des matrices. Une idée consistera à analyser l'emploi d'une moyenne autre (typiquement géométrique ou harmonique) et à étudier son influence sur les résultats de classification.

Ce travail a débouché sur une publication en conférence internationale :

> J. Muzeau, P. Ladret et P. Bertolino, "Linear classification of chair-lift images for presence analysis", International Conference on Quality Control by Artificial Vision, 2019. [Muz+19]

Deuxième partie

Comptage du nombre de passagers

3 Problématique et méthodologie adoptée

Sommaire

3.1 Motivations

Nous nous focalisons dans cette partie sur le comptage des passagers sur chaque véhicule. Cette information est en effet fondamentale pour notre application, principalement pour deux raisons.

D'abord, le nombre de skieurs sur chaque véhicule est un élément clé dans l'analyse du danger. Si aucun passager n'est présent, alors la situation est vraisemblablement sécurisée. Dans le cas d'au moins un passager, il faut surveiller le déséquilibre du siège (créé par deux personnes assises du même côté par exemple), ce dernier pouvant engendrer des problèmes mécaniques et surtout des chutes. De plus, le comptage permet d'éviter une surcharge, qui intervient quand le nombre de skieurs dépasse la capacité maximale d'accueil du siège, évidemment source de danger.

Deuxièmement, ce dénombrement peut aider à l'estimation en temps-réel du flux de skieurs, d'une part sur la remontée mécanique, d'autre part sur l'ensemble du domaine skiable. Dans le premier cas, il est important de pouvoir évaluer le nombre total de personnes présentes sur le télésiège : en cas de panne prolongée et ainsi d'évacuation forcée, cette information permet d'optimiser les secours, que ce soit en termes d'effectif et/ou d'organisation. De plus, l'estimation du flux de skieurs est cruciale pour les gérants de domaine skiable, qui peuvent ainsi obtenir de précieuses statistiques et adapter en conséquence la répartition des opérateurs au fil de la journée, voire même sur une saison complète.

Un dernier point que nous souhaitons souligner est le fait que nos efforts se concentrent sur la recherche d'une méthode la plus **non-supervisée** possible. En effet, comme on a pu le voir en section 1.3, beaucoup de paramètres du système SIVAO sont fixés manuellement et la configuration pour une nouvelle remontée mécanique s'en retrouve complexifiée. Le même problème se pose si on utilise des algorithmes supervisés, c'est-à-dire nécessitant une vérité-terrain chronophage et onéreuse à déterminer, pour apprendre un modèle : la généralisation à n'importe quel télésiège n'est pas évidente.

3.2 Analyse des données fournies

Intéressons-nous dans un premier temps aux données auxquelles nous avons affaire. Il s'agit d'environ 1500 vidéos de deux minutes en moyenne, capturées de manière aléatoire au fil de la saison et au long de la journée. Trois stations et dix-neuf

télésièges sont concernés. En Figure 3.1 est donné un exemple d'enregistrements effectués (points noirs) sur une remontée : on constate en effet qu'ils sont en général étalés sur la journée mais aucun motif évident n'apparaît en ce qui concerne les jours sélectionnés.

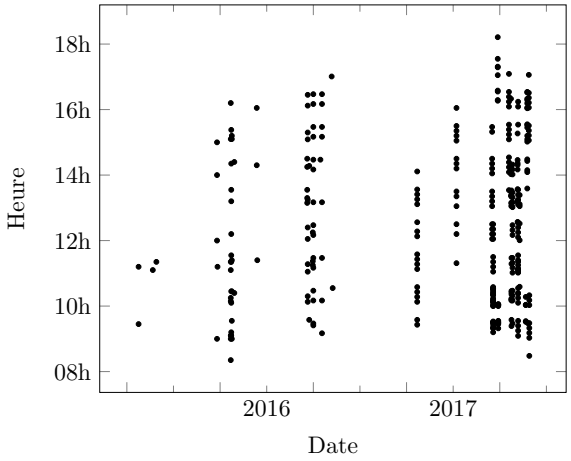


Figure 3.1. Répartition temporelle (date et heure) des vidéos sur une remontée mécanique. Chaque point représente un enregistrement.

Chaque vidéo contient une dizaine de passages de véhicule, nombre qui varie en fonction de la géométrie de la remontée mécanique, mais également en fonction de sa vitesse au cours de la journée. Connaissant la vérité-terrain (c'est-à-dire le nombre réel de passagers assis sur chaque véhicule), on peut analyser sa distribution pour chaque télésiège. Quatre exemples sont donnés en Figure 3.2. Différentes choses sont à noter : d'abord, on remarque que le capacité maximale d'accueil varie d'une remontée à l'autre (quatre et six sur la figure). On observe également une disparité dans le nombre de passages total. Ensuite, les deux graphes de la première ligne reflètent la situation "normale", *i.e.* celle de la plupart des télésièges : une majorité de passages vides, les autres types de véhicules (c'est-à-dire avec un nombre de passagers supérieur ou égal à un) étant répartis quasi-uniformément en général. Toutefois, une remontée ne respecte pas cette distribution (en bas à droite). Enfin, remarquons la valeur -1 en bas à gauche : elle signifie que l'annotation manuelle n'a pas permis de déterminer le nombre de passagers exact. En l'occurence, le problème pour ce télésiège est la présence d'une bulle de protection (contre le vent, la neige, la pluie, le froid, *etc.*) qui bloque la visibilité.

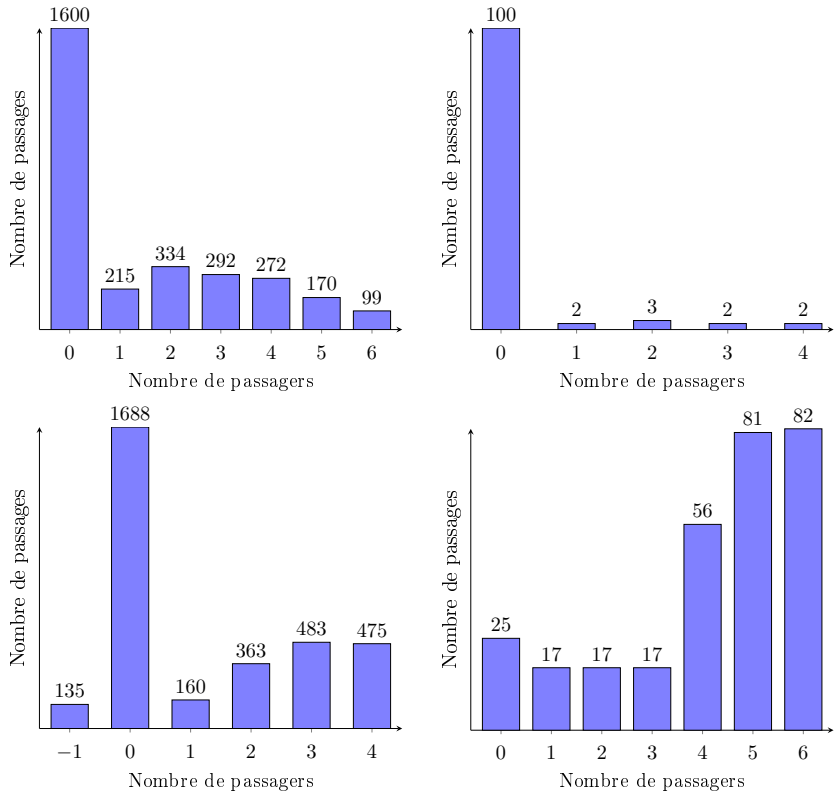


Figure 3.2. Distribution du nombre de passagers pour quatre remontées différentes.

3.3 Pipeline proposé

3.3.1 Présentation synthétique

Nous détaillons dans cette section la méthode proposée, dont le but est le comptage de passagers sur chaque véhicule, et ce de la manière la plus non-supervisée possible. Le pipeline se décompose en trois étapes :

1. D'abord, on procède à une **réduction de dimension**. En effet, du fait de la grande résolution des images unitaires concernées, cette première étape permet de ne conserver que les éléments porteurs d'information et donc de simplifier le problème pour les traitements ultérieurs.

2. Ensuite, une technique de **partitionnement de données** est appliquée. L'idée est de regrouper les images représentant des véhicules avec le même nombre de passagers.

3. Enfin, les résultats obtenus à l'étape précédente ne donnant pas directement le nombre exact de skieurs sur chaque véhicule, une étape d'étiquetage manuel

de chaque groupe précédemment déterminé est nécessaire : on parle alors de **transduction**.

3.3.2 Compléments

Nous donnons dans cette sous-section quelques informations supplémentaires sur la méthode précédente.

Dans un premier temps, le travail qui suit est réalisé sur les images unitaires. Pour rappel, ces dernières sont des vignettes extraites (grâce au système de surveillance développé par BLUECIME) des images issues de la caméra et sont centrées sur le véhicule. Elles ne sont obtenues qu'à certains instants (plus précisément à l'intérieur de la zone de détection telle que présentée en Figure 1.1a) : on récupère alors, en moyenne, une quarantaine d'images d'intérêt par passage, chiffre variant en fonction de la remontée mécanique.

Ensuite, il est important d'exploiter la continuité temporelle apportée par ce projet : le processus détaillé précédemment est appliqué aux images d'une seule et même vidéo. En effet, étant donné que le véhicule est centré dans les images unitaires à chaque instant, une comparaison entre elles permet de s'abstraire de l'information du siège et de se focaliser sur les passagers uniquement. Toutefois, cette comparaison n'est pas valide si les images de départ sont trop dissemblables, ce qui est notamment le cas quand elles sont issues de dates/heures différentes. Afin de s'affranchir de cet effet, nous choisissons des images proches temporellement : ainsi les conditions météorologiques sont considérées les mêmes. Pour résumer, le travail est effectué ici sur un intervalle de temps réduit (quelques minutes tout au plus) : chaque itération est différente, indépendante.

3.3.3 Démarche accompagnée d'un exemple illustratif

On se propose dans cette sous-section d'illuster la démarche présentée par un exemple simple, dans le but d'appréhender le problème sans rentrer trop profondément dans les détails.

Considérons la vidéo dont quelques images unitaires sont données en Figure 3.3. Cette vidéo dure 120 secondes, contient 373 images et neuf passages de véhicule, soit approximativement 41 vignettes par passage. Trois passagers sont transportés pour les passages 2 et 3, cinq pour le passage 4, aucun pour les six autres : nous

avons donc affaire à trois classes (à savoir 0, 3, et 5). Les sorties de chaque étape du pipeline proposé sont représentées graphiquement en Figure 3.4.

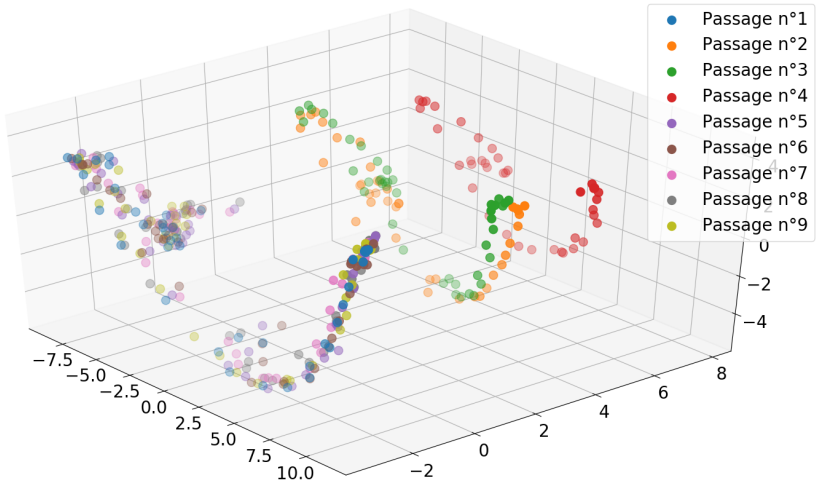
Figure 3.3. Exemples d'images unitaires.

L'étape 1 (à savoir la réduction de dimension) permet d'obtenir, à partir des images unitaires en niveaux de gris en moyenne résolution, un résultat tel que celui présenté en Figure 3.4a. Un espace réduit en trois dimensions est utilisé par simplicité, où chaque point représente une image et où les couleurs permettent de distinguer les différents passages. On peut nettement y voir l'ordonnancement par rapport au nombre de personnes : à gauche les véhicules vides, au milieu les passages avec trois skieurs et à droite le véhicule avec cinq passagers. Ainsi, cette première étape permet de réduire la taille des données tout en gardant l'information essentielle à notre application, c'est-à-dire le regroupement des passages similaires.

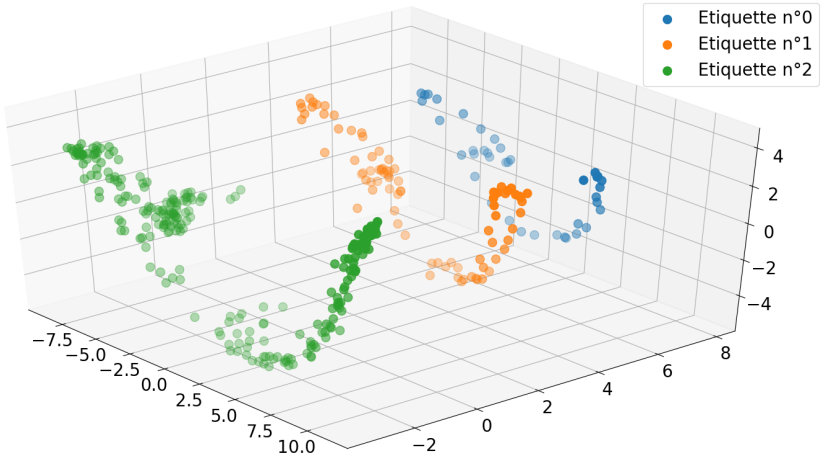
On se propose de récupérer automatiquement ces trois "fers à cheval" via une technique de partitionnement de données (étape 2). Le résultat est donné en Figure 3.4b ainsi que les étiquettes associées. En d'autres termes, cette étape permet de "trier" les passages. On notera que ces étiquettes, construites de manière arbitraire, n'apportent aucune information directe vis-à-vis du nombre de passagers.

Quant à elle, l'étape 3 permet d'obtenir l'information du nombre de passagers. Ce dernier est connu en annotant manuellement une image représentative de chaque groupe (ou *cluster*) déterminé à l'étape précédente. Les annotations choisies sont affichées en rouge sur la Figure 3.4c. Ainsi, l'étiquette 2 (en vert, arbitraire) peut être remplacée par le nombre de passagers donné par le point rouge le plus à gauche, à savoir 0. De même, l'étiquette 1 devient 3 et l'étiquette 0 se transforme en 5.

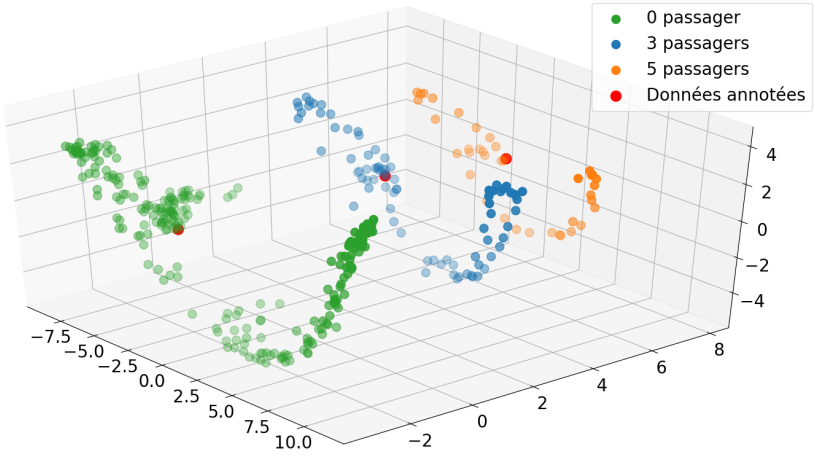
Dans les trois prochains chapitres de ce manuscrit, nous développons les étapes précédentes de manière plus théorique et précisons comment les appliquer en pratique au problème du comptage de passagers de remontées mécaniques.



(a) Images unitaires projetées dans un espace tridimensionnel après réduction de dimension (étape 1).



(b) Différents clusters, en fonction du nombre de passagers présents sur le véhicule, après partitionnement de données (étape 2).



(c) Nombre de skieurs par siège après transduction (étape 3) via annotation manuelle de trois images.

Figure 3.4. Résultats intermédiaires de la chaîne de traitements proposée.

Réduction de dimension

4

Sommaire

4.1 Introduction

Ce chapitre traite d'une étape préliminaire obligatoire à tout processus d'apprentissage automatique et d'exploration de données : la réduction de dimension.

Son but générique est de **réduire le nombre de variables utiles** d'un jeu de données spécifique sans perdre d'information. A titre d'exemple, si l'on considère n données $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ à p variables (ou dimensions), l'objectif de la réduction de dimension est d'obtenir les points transformés $\{\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}\}$ en dimension réduite $d < p$, tels que leur contenu informatif soit similaire à celui des $\boldsymbol{x_i}$.

La réduction de dimension offre de multiples avantages. Le plus évident est sans doute la diminution de la taille du jeu de données et donc de l'occupation en termes de mémoire. De plus, moins de variables à traiter implique une modification de la complexité des algorithmes. Les temps de calcul s'en trouvent ainsi réduits, que ce soit en entraînement ou en test. Enfin, la réduction de la dimensionnalité permet de supprimer les redondances présentes dans les données et d'éliminer le bruit.

Quoi qu'il en soit, la réduction de dimension permet de transformer des données en faible dimension afin de faciliter leur visualisation et compréhension, mais surtout de s'abstraire de ce qu'on appelle la **malédiction de la dimension**. Cette dernière traduit l'idée, formalisée par Richard Bellman en 1961 [Bel03], que le volume de l'espace croît si rapidement avec la dimension que les données s'en trouvent éloignées les unes des autres, rendant n'importe quelle tâche impraticable. On peut intuiter ce phénomène en considérant un carré de côté 1 ainsi que son cercle inscrit. Le rapport des deux aires est égal à :

$$\frac{\mathcal{A}_{\text{cercle}}}{\mathcal{A}_{\text{carré}}} = \frac{\pi \times 0,5^2}{1^2} \approx 0,7854, \tag{4.1}$$

où $\mathcal{A}$ signifie aire. Si l'on transpose maintenant la situation en plus grande dimension, disons dix, ce rapport devient alors :

$$\frac{\mathcal{HV}_{\text{hyper-sphère}}}{\mathcal{HV}_{\text{hyper-cube}}} = \frac{\pi^5 0,5^{10}/\Gamma(6)}{1^{10}} \approx 0,0025, \tag{4.2}$$

où $\mathcal{HV}$ signifie hyper-volume et Γ est la fonction gamma, qui est définie comme suit :

$$\forall z \in \mathbb{C} \text{ tel que } \operatorname{Re}(z) > 0 \qquad \Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} \mathrm{d}t, \tag{4.3}$$

avec la fameuse relation de récurrence $\Gamma(z+1) = z\Gamma(z)$. Le rapport des volumes tend évidemment vers zéro au fur et à mesure que la dimension augmente. Concrètement,

on comprend que l'espace entre l'hyper-cube et l'hyper-sphère croît petit à petit, rendant les points isolés. Une autre manière de sentir ce problème consiste à calculer la distance Euclidienne entre un coin du carré et son centre. Elle vaut $\sqrt{2} \times 0,5$ en deux dimensions, $\sqrt{10} \times 0,5$ pour un espace à dix dimensions : les coins d'un hyper-cube s'éloignent petit à petit de son centre. Cela montre au passage que la notion de distance est troublée en grande dimension. Ce phénomène intervient particulièrement dans tous les pans de l'apprentissage automatique : classification, détection d'anomalies, partitionnemment de données, *etc*. Une solution théorique valide à cette problématique de malédiction de la dimension est l'augmentation du nombre de données nécessaires. On comprend toutefois qu'elle est inenvisageable en pratique du fait de sa complexité, de son coût, sans parler de la détermination de la vérité-terrain associée, elle aussi onéreuse.

Deux manières de procéder à une réduction de dimension existent [Kha+14].

1. La première vise à sélectionner certaines variables (ou *features*) parmi celles originales qui sont les plus pertinentes : on parle de **sélection** de caractéristiques.

2. La deuxième crée de nouvelles dimensions, soit en combinant les variables originales, soit en exploitant la géométrie des données : on parle d'**extraction** de caractéristiques.

On notera également que la majorité des techniques appartenant au domaine de la réduction de dimension sont non-supervisées, c'est-à-dire qu'elles ne nécessitent aucune information *a priori* sur les données (telle que la classe d'appartenance par exemple) pour réduire leur taille.

La réduction de dimension est cruciale pour n'importe quelle application en vision par ordinateur, en particulier la nôtre. En effet, une image de taille 20×20 peut être considérée comme une vecteur dans un espace en 400 dimensions : une si petite image entre déjà dans le cadre de la malédiction de la dimension et engendre les problèmes mentionnés précédemment. De plus, parmi tous les pixels, beaucoup ne sont que peu, voire pas du tout, informatifs (*e.g.* ceux qui concernent l'arrière-plan et qui n'aident pas pour le comptage de passagers) et les supprimer serait bénéfique.

Ce chapitre vise à proposer un état de l'art des différentes techniques de réduction de dimension, en mettant particulièrement l'accent sur celle utilisée dans notre pipeline, à savoir les Cartes de Diffusion (CD).

4.2 Etat de l'art

Cette section se concentre sur les techniques de réduction de dimension par extraction de caractéristiques et catégorise ces dernières en deux groupes : linéaires et non-linéaires. Nous considérons dans la suite un jeu de données avec n individus $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ de p variables que l'on peut ranger ligne par ligne dans la matrice $\boldsymbol{X} = (x_{ij})$ de taille $n \times p$. L'objectif est de réduire ce dernier à d dimensions, avec $d < p$.

4.2.1 Méthodes linéaires

Ce premier ensemble de méthodes vise à **combiner de manière linéaire les différentes variables** afin de réduire la dimensionalité du problème étudié.

Analyse en composantes principales

La technique la plus connue est sans aucun doute l'Analyse en Composantes Principales (ACP). Développée par Karl Pearson en 1901 [Pea01], elle cherche à simplifier le jeu de données en analysant et supprimant les redondances, reflétées pas des corrélations entre variables.

L'ACP peut se résumer par les étapes suivantes :

1. Calculer le vecteur moyen des **variables** $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^T$ tel que

$$\forall j \in [\![1; p]\!] \qquad \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}. \tag{4.4}$$

2. Centrer les variables originales par

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} - \boldsymbol{1_n}\, \boldsymbol{\mu}^T, \tag{4.5}$$

   où $\boldsymbol{1_n}$ est un vecteur de taille $n \times 1$ rempli de uns. Une normalisation par rapport à l'écart-type de chaque variable est également possible.

3. Déterminer la matrice de covariance de $\tilde{\boldsymbol{X}}$:

$$\boldsymbol{C} = \frac{1}{n-1} \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{X}}. \tag{4.6}$$

Le lecteur notera la division par $n-1$ (au lieu de n), appelée correction de Bessel [Ken64], qui permet d'obtenir une estimation non-biaisée de la matrice de covariance.

4. Diagonaliser $\boldsymbol{C}$ tel que
$$\boldsymbol{C} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}, \tag{4.7}$$
où $\boldsymbol{D}$ est une matrice diagonale dont les éléments sont les valeurs propres et $\boldsymbol{P}$ contient les vecteurs propres associés, colonne par colonne.

5. Obtenir $\tilde{\boldsymbol{P}}$ en réordonnant $\boldsymbol{P}$ selon les valeurs propres décroissantes.

6. Garder les d premières colonnes de $\tilde{\boldsymbol{P}}$: ce sont les axes principaux.

La matrice $\boldsymbol{U}$ obtenue, de taille $p \times d$, permet ainsi de projeter les données originales en dimension p dans un espace réduit en dimension d. Précisons que la dimension réduite peut être déterminée empiriquement de différentes manières : la plus utilisée consiste à analyser le pourcentage de variance expliquée défini par la somme cumulative des valeurs propres.

Apportons ici quelques précisions. Comme imaginé originalement par Karl Pearson [Pea01], la matrice $\boldsymbol{U}$ est en fait la solution d'un problème d'optimisation visant à minimiser l'erreur de projection une fois l'espace réduit construit. Plus précisément :

$$\boldsymbol{U} = \underset{\boldsymbol{Q}}{\operatorname{argmin}}\, \mathbb{E}\, \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{Q}\boldsymbol{Q}^T\|_F^2, \tag{4.8}$$

où $\|.\|_F$ est la norme de Frobenius et $\mathbb{E}$ est l'espérance mathématique. Ainsi, l'ACP détermine l'hyper-plan dans l'espace en dimension d qui permet une perte d'information minimale une fois les données originales projetées.

En 1933, Harold Hotelling a démontré que ce problème de minimisation de l'erreur de projection est en fait équivalent à celui de la maximisation de la variance des données projetées [Hot33]. Ces deux points de vues coexistent et aboutissent au même résultat.

En plus de permettre une visualisation et une analyse rapides de n'importe quel jeu de données, l'ACP a été utilisée avec succès dans de nombreux domaines tels que la compression d'images [PM18], la reconnaissance faciale [TP91] ou la séparation premier-plan/arrière-plan dans les vidéos [Oli+00]. Cette méthode est donc extrêmement simple à mettre en oeuvre et permet d'inclure une nouvelle donnée facilement via l'utilisation de la matrice de projection déterminée précédemment. L'inconvénient majeur est évidemment sa linéarité, hypothèse qui peut être coûteuse dans de nombreuses applications réelles.

Analyse discriminante linéaire

Il a été dit en début de section 4.1 que la majorité des techniques de réduction de dimension étaient non-supervisées. Toute règle ayant son exception, l'Analyse Discriminante Linéaire (ADL) [Fis36b] ne fait pas partie de cet ensemble de méthodes. Précisons d'abord que nous parlons ici de l'ADL en tant que processus de réduction de dimension qui, même si elle présente des idées similaires, ne doit pas être confondue avec l'analyse discriminante pour la classification développée en section 2.2. Dans le cas présent, l'information d'appartenance de chaque donnée à une classe spécifique est mise à profit pour la réduction de dimension. Le but de l'ADL est ici de déterminer un nouvel espace réduit qui, une fois les données originales projetées dans ce dernier, engendre une **séparation optimale des différentes classes** : on cherche donc à minimiser la variance intra-classe et à maximiser la variance inter-classe. Concrètement, cela signifie que l'objectif est de trouver un espace (dont, pour rappel, les axes sont des combinaisons linéaires des caractéristiques de base) qui permet de "regrouper" les données d'une même classe tout en séparant au mieux celles des différentes classes.

Le jeu de données $\boldsymbol{X}$ est séparé en deux classes $\mathcal{C}_0$ (n_0 points) et $\mathcal{C}_1$ (n_1 points), avec $n_0 + n_1 = n$. Nous nous concentrons ici sur le cas de deux classes par simplicité, le raisonnement peut toutefois être généralisé à n'importe quel nombre de classes. On peut décomposer le processus de réduction de dimension par ADL comme suit :

1. Calculer la variance intr**a**-classe

$$\boldsymbol{S_a} = \sum_{\boldsymbol{x_i} \in \mathcal{C}_0} (\boldsymbol{x_i} - \boldsymbol{\mu_0})(\boldsymbol{x_i} - \boldsymbol{\mu_0})^T + \sum_{\boldsymbol{x_i} \in \mathcal{C}_1} (\boldsymbol{x_i} - \boldsymbol{\mu_1})(\boldsymbol{x_i} - \boldsymbol{\mu_1})^T, \tag{4.9}$$

   où $\boldsymbol{\mu_0}, \boldsymbol{\mu_1} \in \mathbb{R}^p$ sont les centres de chaque classe. L'idée sous-jacente à cette matrice est d'évaluer à quel point chaque classe est éparse vis-à-vis de son centre.

2. Déterminer la variance inte**r**-classe

$$\boldsymbol{S_r} = n_0(\boldsymbol{\mu} - \boldsymbol{\mu_0})(\boldsymbol{\mu} - \boldsymbol{\mu_0})^T + n_1(\boldsymbol{\mu} - \boldsymbol{\mu_1})(\boldsymbol{\mu} - \boldsymbol{\mu_1})^T, \tag{4.10}$$

   où $\boldsymbol{\mu} \in \mathbb{R}^p$ est le centre du jeu de données complet. Cette matrice caractérise l'éloigement des classes l'une de l'autre.

3. Obtenir les vecteurs et valeurs propres de $\boldsymbol{S_a}^{-1}\boldsymbol{S_r}$. Comme pour l'ACP, en réorganisant les vecteurs propres selon l'ordre décroissant des valeurs propres, les axes les plus significatifs permettent de regrouper les données projetées

d'une même classe (via $\boldsymbol{S}_a^{-1}$) tout en éloignant au maximum les différentes classes (grâce à $\boldsymbol{S}_r$).

L'ADL en tant que technique de réduction de dimension a notamment été utilisée en reconnaissance faciale par ordinateur via les *Fisherfaces* [Bel+97] dont l'idée est d'extraire, à partir d'images de visages, des caractéristiques discriminantes permettant par la suite une classification aisée. L'ADL est intéressante de par sa simplicté à mettre en oeuvre ainsi que sa linéarité. C'est également un de ses points faibles car cette méthode donne de mauvais résultats dans le cas de situations plus complexes. De surcroît, il faut souligner que l'analyse discriminante linéaire utilise l'hypothèse de normalité, c'est-à-dire le fait que chaque classe est distribuée selon une loi de Gauss, qui s'avère être forte dans de nombreux cas réels. Enfin, rappelons que cette méthode est supervisée, ainsi l'utilisation d'une vérité-terrain est nécessaire.

4.2.2 Méthodes non-linéaires

Contrairement aux algorithmes détaillés précédemment, les techniques données dans cette sous-section ne cherchent pas un espace réduit combinaison linéaire des axes originaux. La réduction **non-linéaire** de la dimensionalité est étroitement liée au concept d'apprentissage de variété mathématique. L'idée sous-jacente est la suivante : les données observées ne sont qu'artificiellement en grande dimension et leur véritable structure repose sur une variété, au sens mathématique du terme, en faible dimension.

Variété mathématique

Par définition, une variété est un espace topologique localement Euclidien, c'est-à-dire dans lequel il est possible de définir, en chaque point, une notion de distance et d'angle. Un exemple typique est la Terre : sphère dans $\mathbb{R}^3$, elle peut aisément être décomposée localement en plans bidimensionnels, sans perte d'information. Le cercle dans le plan est également un exemple de 1-variété (*i.e.* à une dimension) : on n'a ainsi besoin que d'une seule variable pour se "déplacer" sur le cercle.

Ces méthodes sont particulièrement intéressantes pour notre application. En effet, les images d'une même vidéo sont assez similaires, paire à paire. On peut donc supposer que la transformation entre deux images consécutives est linéaire. Ainsi, elles reposent toutes sur une variété mathématique : l'utilisation d'algorithmes de réduction de dimension non-linéaires est donc adaptée à notre projet.

La réduction de dimension non-linéaire est un domaine actif de recherche et de nombreuses techniques ont pu voir le jour ces vingt dernières années. Parmi les plus connues, on peut citer :

- **t-distributed Stochastic Neighbor Embedding (t-SNE)** [MH08 ; Maa14]
- **Uniform Manifold Approximation and Projection (UMAP)** [McI+18]
- **Multi-Dimensional Scaling (MDS)** [Kru64a ; Kru64b ; BG05]
- **Local Tangent Space Alignment (LTSA)** [ZZ02]
- **Isomap** [Ten+00]

Nous détaillons dans la suite de cette sous-section cinq méthodes principales de réduction de dimension par apprentissage de variété, à savoir **(1)** Analyse en Composantes Principales (ACP) à noyau [Sch+98], **(2)** Locally-Linear Embedding (LLE) [RS00 ; SR01], **(3)** Laplacian eigenmaps [BN03], **(4)** Auto-Encodeur (AE) [Kra91]., et **(5)** Cartes de Diffusion (CD) [CL06 ; Tal+13]

Analyse en composantes principales à noyau

L'ACP à noyau (*kernel-PCA* en anglais) est une généralisation de l'analyse en composantes principales pour des problèmes non-linéaires. L'élément important de cette méthode est l'usage de l'**astuce du noyau** [Aiz+64], détaillée en Appendice B.

L'analyse en composantes principales linéaire fonctionne par la décomposition en valeurs et vecteurs propres de la matrice de covariance des données (voir sous-section 4.2.1). L'ACP à noyau procède de la même manière, excepté le fait que la matrice de covariance est calculée dans l'espace d'arrivée d'une fonction de transformation φ de l'astuce du noyau. Considérons le jeu de données $\boldsymbol{X}$ et une fonction de transformation $\varphi : \mathbb{R}^p \to \mathbb{R}^D$. On peut alors exprimer la matrice de covariance dans $\mathbb{R}^D$ par :

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} \varphi(\boldsymbol{x_i}) \varphi(\boldsymbol{x_i})^T. \tag{4.11}$$

Par la suite, cette matrice est diagonalisable et on peut déterminer des valeurs propres λ et vecteurs propres $\boldsymbol{v}$ tels que :

$$\lambda \boldsymbol{v} = \boldsymbol{C}\boldsymbol{v}. \tag{4.12}$$

On peut déduire de cette dernière équation que $\boldsymbol{v}$ appartient au sous-espace vectoriel engendré par les données transformées $\varphi(\boldsymbol{x_1}), \ldots, \varphi(\boldsymbol{x_n})$. Il est ainsi possible de déterminer des coefficients α_i tels que :

$$\boldsymbol{v} = \sum_{i=1}^{n} \alpha_i \, \varphi(\boldsymbol{x_i}). \tag{4.13}$$

Schölkopf *et al.* ont montré dans [Sch+98] que, afin de déterminer ces coefficients, il suffisait de résoudre le problème aux valeurs propres suivant :

$$n\lambda\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{\alpha}, \tag{4.14}$$

où $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$ et où $\boldsymbol{K}$ est la matrice du noyau, autrement dit :

$$\boldsymbol{K} = (k(\boldsymbol{x_i}, \boldsymbol{x_j})) = (\langle \varphi(\boldsymbol{x_i}), \varphi(\boldsymbol{x_j}) \rangle) \quad \text{avec} \quad i, j \in \{1, \ldots, n\}. \tag{4.15}$$

Après une normalisation des vecteurs propres $\boldsymbol{\alpha}$, il reste simplement à calculer la projection d'un nouveau point test transformé $\varphi(\boldsymbol{x})$ sur chacun des vecteurs propres $\boldsymbol{v}$, associé à une valeur propre strictement positive. Ainsi, pour chaque l, on utilise :

$$\langle \boldsymbol{v_l}, \varphi(\boldsymbol{x}) \rangle = \sum_{i=1}^{n} \alpha_{i,l} \langle \varphi(\boldsymbol{x_i}), \varphi(\boldsymbol{x}) \rangle. \tag{4.16}$$

On appelle la partie de gauche la $l^{\text{ème}}$ composante principale non-linéaire.

L'algorithme de l'analyse en composantes principales à noyau peut donc se résumer en trois étapes :

1. Déterminer la matrice du noyau $\boldsymbol{K}$.
2. Diagonaliser $\boldsymbol{K}$ et normaliser les vecteurs propres résultants.
3. Projeter une nouvelle donnée sur les vecteurs propres.

Nous souhaitons souligner que le développement théorique (Équation 4.11 à Équation 4.13) n'entre pas directement en jeu mais est nécessaire afin de comprendre l'algorithme de l'ACP à noyau.

Notons que le raisonnement précédent suppose que les données transformées soient centrées. Une modification des équations est toutefois détaillée dans [Sch+98] afin de prendre en compte le cas de données non-centrées.

L'ACP à noyau a été utilisée avec succès dans les domaines de la reconnaissance faciale et des modèles de formes actives [Wan12]. Un inconvénient majeur de cette méthode est le choix du noyau ainsi que des paramètres associés, souvent dépendants de l'application.

Locally-linear embedding

Cet algorithme, proposé par Roweis et Saul en 2000, provient d'une intuition géométrique simple : une donnée reposant sur une variété mathématique peut être approchée par ses voisins de manière linéaire, il en va de même pour cette donnée projetée dans un espace réduit.

La **première étape** consiste à déterminer les k plus proches voisins de chaque point $\boldsymbol{x_i}$, via la distance Euclidienne classique. **Deuxièmement**, il faut, pour chaque point, déterminer les poids qui permettent d'obtenir la meilleure approximation de ce dernier grâce à ses voisins. En d'autres mots, la fonction de coût à minimiser est la suivante :

$$E(\boldsymbol{W}) = \sum_i \|\boldsymbol{x_i} - \sum_j W_{ij}\boldsymbol{x_j}\|_2^2, \tag{4.17}$$

avec W_{ij} le poids associé à la donnée j afin d'approcher la donnée i. A noter que la deuxième somme s'effectue uniquement sur les k plus proches voisins de $\boldsymbol{x_i}$. On peut comprendre cette équation comme étant l'erreur de reconstruction de chaque point. On remarque également, comme dit précédemment, que cette approximation est linéaire. La solution de ce problème d'optimisation peut facilement être résolue par les moindres carrés, les poids sont ensuite normalisés de telle sorte que la somme ligne par ligne soit égale à 1. La **troisième et dernière étape** vise à trouver des nouvelles coordonnées $\{\boldsymbol{y_1}, \dots, \boldsymbol{y_n}\}$ qui préservent la notion de voisinage définie par les poids $\boldsymbol{W}$. En l'occurence, on cherche à minimiser :

$$E'(\boldsymbol{Y}) = \sum_i \|\boldsymbol{y_i} - \sum_j W_{ij}\boldsymbol{y_j}\|_2^2. \tag{4.18}$$

Notons que cette fonction de coût ressemble à celle de l'Équation 4.17 : toutefois, l'optimisation s'effectue ici sur les $\boldsymbol{y_i}$ alors que les poids W_{ij} sont fixés. Saul et Roweis montrent dans [SR01] que la solution de ce problème peut être déterminée par les vecteurs propres de la matrice $\boldsymbol{M} = (M_{ij})$ telle que

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki}W_{kj}, \tag{4.19}$$

où δ_{ij} est le delta de Kronecker ($\delta_{ij} = 1$ si $i = j$, 0 sinon).

Parmi les applications de l'algorithme locally-linear embedding, on retrouve l'ordonnancement d'images de lèvres dans le cadre de la synthèse de discours audio-visuels [SR01] et la classification pour la maladie d'Alzheimer grâce à l'imagerie à résonnance magnétique [Liu+13a]. Le principal défaut de LLE est l'apparition éventuelle

de différentes instabilités dues aux deux processus d'optimisation : pour pallier ces difficultés, des modifications à base de régularisation ont été proposées par Donoho et Grimes [DG03], ainsi que par Zhang et Wang [ZW06].

Laplacian eigenmaps

Ce sont Belkin et Niyogi qui ont proposé l'algorithme des Laplacian eigenmaps en 2003 [BN03]. Ici, c'est l'idée que les données, appartenant à une variété mathématique, forment un graphe qui est exploitée.

La technique proposée se décompose en trois étapes :

1. **Construire le graphe d'adjacence**
   On considère une liaison entre deux données $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ si ces dernières sont voisines, soit via l'algorithme des k plus proches voisins, soit par une notion de voisinage définie par une distance maximale ε.

2. **Déterminer les poids du graphe**
   On peut simplement fixer le poids de l'arrête entre $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ à 1 si ces deux points sont voisins, suivant le sens défini à l'étape précédente. Une deuxième solution consiste à appliquer un noyau, typiquement Gaussien, de telle manière que :
   $$W_{ij} = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}{2\sigma^2}\right), \tag{4.20}$$
   où W_{ij} est le poids de l'arrête entre $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ s'ils sont connectés, 0 sinon. A noter que ce choix implique l'ajout du paramètre σ qui peut être difficile à régler en pratique.
   A ce stade, nous avons construit une graphe où chaque noeud représente une donnée d'origine $\boldsymbol{x_i}$. De plus, supposons dans la suite que ce graphe est connecté, c'est-à-dire que chaque noeud est relié à tous les autres, que ce soit de manière directe ou indirecte.

3. **Calculer les eigenmaps**
   Suite à la détermination de la matrice des poids $\boldsymbol{W}$, il est possible de calculer les matrices $\boldsymbol{D}$ et $\boldsymbol{L}$ telles que :
   $$D_{ii} = \sum_j W_{ij}. \tag{4.21}$$
   $$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}. \tag{4.22}$$

La matrice $\boldsymbol{D}$ est diagonale dont les éléments sont la somme des poids ligne par ligne. Quant à $\boldsymbol{L}$, elle porte le nom de matrice **Laplacienne**, d'où le nom de la méthode. On cherche par la suite à résoudre le problème aux valeurs propres généralisé :

$$\boldsymbol{L}\boldsymbol{v} = \lambda \boldsymbol{D}\boldsymbol{v}, \tag{4.23}$$

où λ sont les valeurs propres et $\boldsymbol{v}$ les vecteurs propres. En ordonnant les n vecteurs propres de manière à ce que les valeurs propres soient décroissantes, il suffit alors de choisir les d premiers afin d'obtenir le nouvel espace réduit en dimension d. Notons également que la première valeur propre est égale à 0, résultat classique de la théorie des graphes : le vecteur propre associé peut donc être mis de côté.

Cette méthode présente des propriétés intéressantes : la plus importante sûrement est l'analogie entre la matrice Laplacienne et l'opérateur différentiel Laplacien. En l'occurence, la matrice Laplacienne est une approximation de l'opérateur de Laplace-Beltrami sur les variétés mathématiques. De plus, l'algorithme proposé induit une préservation du voisinage local, c'est-à-dire que des données proches dans l'espace original le sont après réduction.

Belkin et Niyogi proposent dans [BN03] une application dans le domaine de la reconnaissance automatique de la parole. En exécutant la méthode des Laplacian eigenmaps à la transformée de Fourier à court terme d'un signal de parole, les auteurs obtiennent une représentation en deux dimensions permettant de regrouper les sons similaires.

Auto-encodeur

L'apprentissage profond a pu se développer ces dernières années notamment grâce à l'amélioration du potentiel de calcul des ordinateurs. Une réduction de dimension est donc aujourd'hui possible par les Auto-Encodeurs (AE).

Une représentation schématique d'un AE est donnée en Figure 4.1. On se place dans un cadre simplifié avec une couche d'entrée (en vert), une couche cachée (en rouge) et une couche de sortie (en bleu). Les entrée $\boldsymbol{X}$ et sortie $\tilde{\boldsymbol{X}}$ sont en dimension p, alors que la couchée cachée $\boldsymbol{Y}$ est en dimension d. Le principe général consiste à entraîner un réseau de neurones, via rétro-propagation [Rum+86], de telle manière que la sortie soit identique à l'entrée. La première partie constitue

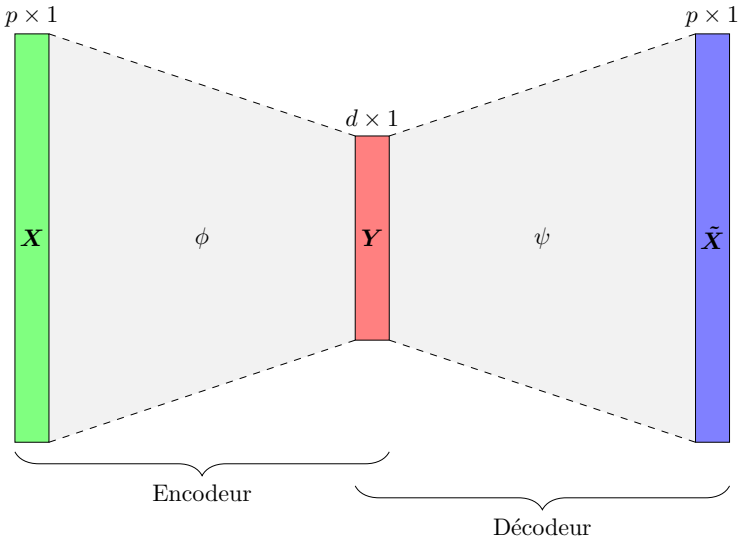


Figure 4.1. Schéma simplifié du fonctionnement d'un auto-encodeur. Les couches d'entrée, cachée et de sortie sont respectivement représentées en vert, rouge et bleu.

l'**encodeur** représenté par la fonction ϕ, la deuxième est appelée **décodeur**, dénoté par ψ. On cherche donc à déterminer ϕ et ψ telles que :

$$\phi, \psi = \underset{f,g}{\operatorname{argmin}} \|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_2^2 = \underset{f,g}{\operatorname{argmin}} \|\boldsymbol{X} - (g \circ f)\boldsymbol{X}\|_2^2. \tag{4.24}$$

La couche cachée $\boldsymbol{Y}$, en dimension $d < p$, constitue donc la "projection" d'une donnée $\boldsymbol{X}$: ce sont les variables latentes.

Les AE permettent d'extraire plusieurs niveaux de caractéristiques et donc d'obtenir une représentation approfondie des données. Un inconvénient majeur de cette technique est le fait que l'architecture conditionne la dimension réduite : elle est fixée avant entraînement et n'est pas adaptative par rapport aux données en entrée.

Cartes de diffusion

Les Cartes de Diffusion (CD) nous intéressent particulièrement car elles seront utilisées dans la suite de ce manuscrit. Le cadre théorique de cette méthode a été formulé par Coifman et Lafon en 2006 [CL06]. Son idée repose, tout comme les Laplacian eigenmaps, sur le fait que les données constituent un graphe dont elles sont les noeuds. Cependant, la réduction de dimension se fait, contrairement à

d'autres techniques, en pratiquant une marche aléatoire [Pea05] sur ce graphe considéré alors comme une chaîne de Markov.

On peut résumer l'algorithme des cartes de diffusion comme suit :

1. **Calculer les distances paire à paire**
   Il faut d'abord définir une notion de distance (ou similarité) entre les données. Typiquement, on peut choisir la distance Euclidienne.
2. **Déterminer la matrice noyau**

$$\boldsymbol{K} = (k_{ij}) \quad \text{où} \quad k_{ij} = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}{2\sigma^2}\right). \tag{4.25}$$

   Les éléments de cette matrice tendent vers 1 si les données considérées sont proches au sens de la distance définie à l'étape précédente, ils tendent vers 0 dans le cas contraire.
3. **Normaliser la matrice noyau**

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{K}, \tag{4.26}$$

   où $\boldsymbol{D}$ est la matrice diagonale dont les éléments sont la somme ligne par ligne de la matrice noyau, c'est-à-dire $D_{ii} = \sum_j K_{ij}$. On appelle $\boldsymbol{P}$ la matrice Markovienne de transition. Remarquons que cette normalisation implique que toutes les valeurs de cette dernière sont comprises entre 0 et 1 et que la somme ligne par ligne est égale à 1. Ainsi, l'élément p_{ij} peut être vu comme la probabilité de se déplacer de $\boldsymbol{x_i}$ vers $\boldsymbol{x_j}$.
4. **Décomposition en valeurs singulières de la matrice de transition**
   On obtient un ensemble de valeurs singulières $\{\lambda_1, \ldots, \lambda_n\}$ triées dans l'ordre décroissant et les vecteurs singuliers associés gauches $\{\boldsymbol{\varphi_1}, \ldots, \boldsymbol{\varphi_n}\}$ et droits $\{\boldsymbol{\psi_1}, \ldots, \boldsymbol{\psi_n}\}$. Notons que les valeurs singulières sont également comprises entre 0 et 1 et que $\lambda_1 = 1$, impliquant un vecteur singulier $\boldsymbol{\psi_1}$ proportionnel au vecteur unitaire.
5. **Construire l'espace réduit**
   Pour une donnée $\boldsymbol{x} \in \mathbb{R}^p$, on peut déduire sa transformation $\boldsymbol{\Psi_t}(\boldsymbol{x})$ dans l'espace réduit en dimension d par :

$$\boldsymbol{\Psi_t}(\boldsymbol{x}) = \left(\lambda_2^t \boldsymbol{\psi_2}(\boldsymbol{x}), \ldots, \lambda_{d+1}^t \boldsymbol{\psi_{d+1}}(\boldsymbol{x})\right). \tag{4.27}$$

   Une interprétation du paramètre t est donnée dans la suite.

Un des premiers éléments que l'on peut noter est la ressemblance des cartes de diffusion avec les Laplacian eigenmaps (formation d'un graphe par les données et

diagonalisation notamment). Toutefois, les CD se distinguent du fait de l'information apportée par le caractère Markovien (c'est-à-dire les valeurs propres à la puissance t dans l'Équation 4.27). Talmon *et al.* [Tal+13] soulignent d'ailleurs que les deux algorithmes coïncident si cet apport est mis de côté.

Revenons un instant sur l'interprétation de la matrice Markovienne $\boldsymbol{P}$ et du paramètre t. Comme décrit ci-dessus, chaque élément p_{ij} correspond à la probabilité de se déplacer vers le noeud $\boldsymbol{x_j}$ sachant que nous sommes sur le noeud $\boldsymbol{x_i}$. On se doit de préciser que cette transition se fait sur un intervalle de temps. Concrètement, on a :

$$p_{ij} = \mathbb{P}(\boldsymbol{x_j} \text{ à l'instant } T+1 | \boldsymbol{x_i} \text{ à l'instant } T). \tag{4.28}$$

Cette transition peut se faire sur une durée plus longue, disons t étapes. Dans ce cas, il suffit de prendre la matrice Markovienne à la puissance t. On obtient ainsi :

$$p_{ij}^{(t)} = \mathbb{P}(\boldsymbol{x_j} \text{ à l'instant } T+t | \boldsymbol{x_i} \text{ à l'instant } T), \tag{4.29}$$

où $p_{ij}^{(t)}$ représente l'élément placé à la ligne i et à la colonne j de la matrice $\boldsymbol{P}^t$. Précisons que la puissance évoquée ici est bien matricielle, non pas élément par élément. Cette évolution temporelle se retrouve à l'Équation 4.27 via la mise à la puissance t des valeurs propres.

Une autre propriété de ces cartes de diffusion est la déduction d'une nouvelle mesure d'affinité entre deux données originales $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$. Cette métrique, appelée **distance de diffusion**, s'exprime ainsi :

$$D_t^2(\boldsymbol{x_i}, \boldsymbol{x_j}) = \|p_{i.}^{(t)} - p_{j.}^{(t)}\|_{\boldsymbol{\varphi_1}}^2 \tag{4.30}$$

$$= \sum_{k=1}^{n} \frac{1}{\boldsymbol{\varphi_1}(k)} (p_{ik}^{(t)} - p_{jk}^{(t)})^2 \tag{4.31}$$

où $\boldsymbol{\varphi_1}(k)$ désigne la $k^{\text{ème}}$ composante du vecteur $\boldsymbol{\varphi_1}$. De cette manière, la distance entre deux données prend en compte l'information globale du graphe défini, c'est-à-dire qu'elle dépend de tous les autres points : même au niveau local, l'information globale entre en jeu. De plus, Coifman et Lafon [CL06] montrent que cette distance de diffusion est égale à la distance Euclidienne dans l'espace transformé si on considère l'ensemble des n valeurs et vecteurs singuliers ; on en obtient toutefois une bonne approximation en ne gardant que les d premiers valeurs et vecteurs singuliers. Autrement dit :

$$D_t^2(\boldsymbol{x_i}, \boldsymbol{x_j}) \approx \|\boldsymbol{\Psi_t}(\boldsymbol{x_i}) - \boldsymbol{\Psi_t}(\boldsymbol{x_j})\|^2. \tag{4.32}$$

Nous nous proposons d'illustrer cette méthode sur un exemple simple. Considérons une vidéo d'un "pendule" oscillant [1] dont quelques images sont données en Figure 4.2. Bien que les données soient en grande dimension, il est possible d'appliquer l'algorithme des CD et d'obtenir une réduction à une seule dimension, le résultat est affiché en Figure 4.3. On constate sur cette figure qu'un ordre entre les images s'est établi : en effet, les points les plus à gauche (en bleu marine) représentent des images dans lesquelles le pendule est incliné vers la gauche et réciproquement pour les points situés sur la droite (en rouge foncé). Quant à ceux au milieu, ils correspondent à la situation où le pendule est en position verticale. Ainsi, les CD permettent, de manière générale, de récupérer les paramètres sous-jacents qui ont abouti à la génération des données de la scène analysée : en l'occurence, dans cet exemple simple, le seul paramètre qui varie au cours du temps est l'angle d'inclinaison du pendule, élément que l'on retrouve via la réduction de dimension.

4.3 Application aux images de télésièges

Après avoir évoqué l'état de l'art dans le domaine de la réduction de dimension, nous proposons dans cette section d'appliquer les Cartes de Diffusion (CD) au problème de la sécurité des passagers sur les remontées mécaniques. En particulier, nous verrons comment cette méthode peut nous aider au comptage des skieurs sur chaque véhicule.

Comme mentionné en sous-section 3.3.2, rappelons que nous travaillons vidéo par vidéo. En appliquant ainsi les CD sur un intervalle de temps assez court, c'est-à-dire la durée de chaque vidéo (soit deux minutes environ), on s'abstrait au maximum des variabilités du milieu extérieur, *e.g.* les variations d'illumination dues à la météo : la comparaison entre images permet par la suite de se concentrer sur les éléments intéressants, à savoir les passagers. De plus, on rappelle que ce sont les images unitaires qui sont utilisées, c'est-à-dire des vignettes centrées sur le siège à chaque passage d'un véhicule.

Pour chaque vidéo, nous prétraitons les données comme suit :

1. Les images sont d'abord converties en niveaux de gris.
2. Ces images unitaires étant originalement carrées, elles sont ensuite redimensionnées en 50 pixels de haut et 50 pixels de large. Ces valeurs sont déterminées empiriquement car aboutissent aux meilleurs résultats.

1. Merci à Pedro Luiz COELHO RODRIGUES pour les images.

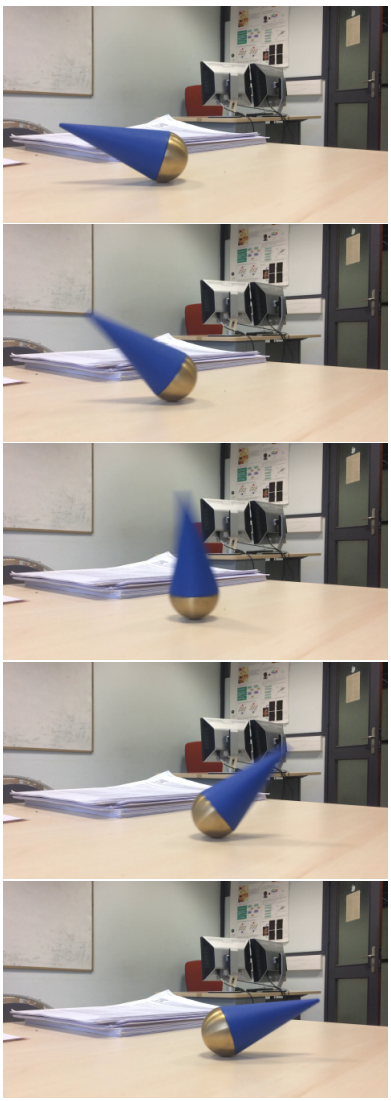


Figure 4.2. Images extraites d'une vidéo d'un pendule qui oscille.

3. Puis, on procède à une normalisation des niveaux de gris vers l'intervalle $[0\,;\,1]$.
4. Enfin, on transforme les données résultantes sous la forme de vecteurs de taille $50 \times 50 = 2500$.

Ensuite, nous appliquons l'algorithme des cartes de diffusion à ces vecteurs. Nous fixons t, le temps de transition lié à la matrice Markovienne, à 1 et l'écart-type σ de la fonction noyau (voir Équation 4.25) égal à la médiane des distances entre images, paire à paire. La distance considérée ici est Euclidienne, c'est-à-dire que les images sont transformées en vecteurs via concaténation des colonnes puis la différence est calculée sur les niveaux de gris pixel à pixel. Nous nous limitons pour le moment à une réduction à trois dimensions afin de faciliter la visualisation des données et de

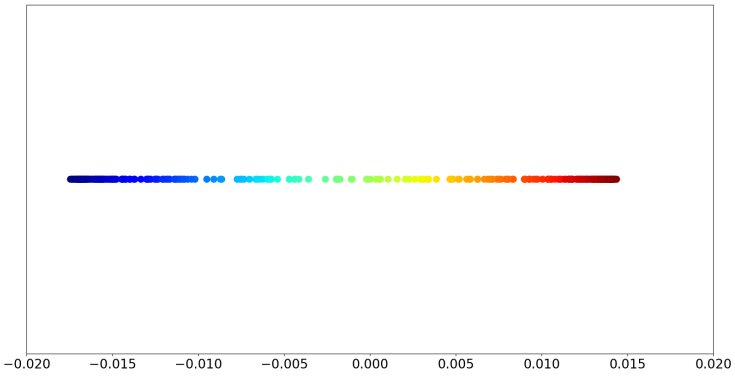


Figure 4.3. Cartes de diffusion appliquées à la vidéo du pendule. Réduction à une seule dimension. Un ordre en fonction de l'inclinaison du pendule s'est établi.

comprendre ce qui se joue : dans la suite de ce manuscrit, la dimension réduite sera déterminée automatiquement.

Six résultats sont donnés en Figure 4.4. Chaque point représente une image unitaire réduite en dimension. L'information d'appartenance à tel ou tel passage n'est pas donnée ici, contrairement à la Figure 3.4a par exemple : les couleurs correspondent aux "types" de véhicules de la vidéo, c'est-à-dire aux différents nombres de passagers présents sur les sièges.

L'élément fondamental à noter sur cette figure est l'apparition d'un ordre parmi les images, **ordre gouverné par le nombre de passagers de chaque véhicule**. Sur la Figure 4.4a, tous les sièges sont vides : les différents points sont ainsi connectés et forment un unique "bloc". En ce qui concerne les trois figures suivantes (Figure 4.4b, Figure 4.4c et Figure 4.4d), les véhicules transportant le même nombre de skieurs sont regroupés et la séparation avec les autres types de sièges est claire. Enfin, bien qu'un motif se dessine, la situation n'est pas aussi évidente sur les deux derniers exemples. En effet, certaines transitions sont plutôt floues : c'est le cas de celles liées aux données en vert dans la Figure 4.4e et de celle entre les véhicules avec deux ou trois passagers dans la Figure 4.4f.

4.4 Conclusion

Nous avons présenté dans ce chapitre le principe de la réduction de dimension, ses avantages et son intérêt dans la problématique de la sécurité des personnes sur les télésièges. Nous avons ensuite procédé à un état de l'art de ce domaine,

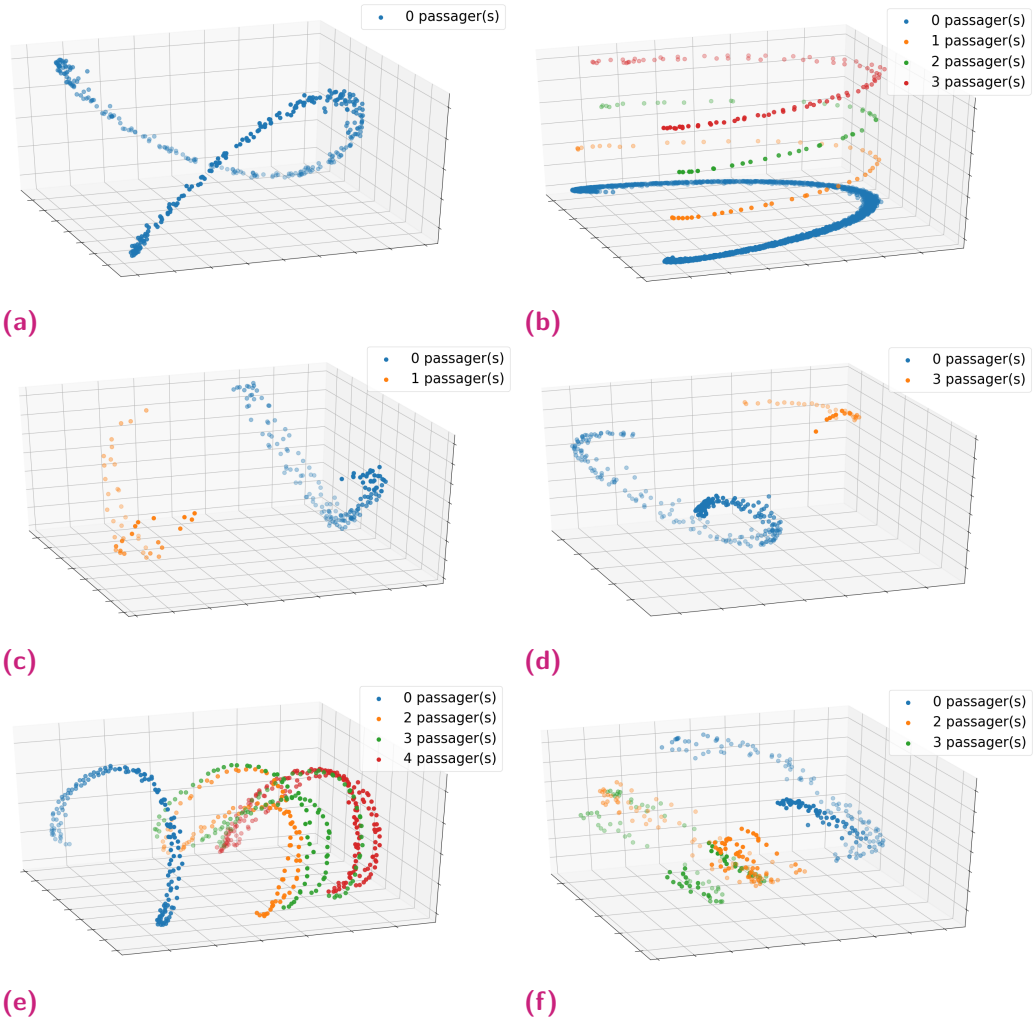


Figure 4.4. Exemples de l'algorithme des cartes de diffusion appliqué aux images unitaires extraites de vidéos de rémontées mécaniques. La séparation en fonction du nombre de passagers présents sur chaque véhicule est discernable.

en distinguant les méthodes linéaires de celles par apprentissage de variété, non-linéaires. L'accent a été mis sur l'algorithme des Cartes de Diffusion (CD).

Nous avons pu voir que la technique des CD aboutit à une séparation des images unitaires d'une vidéo en fonction du nombre de passagers présents sur chaque véhicule. Rappelons que cette méthode de réduction de dimension est complètement non-supervisée, c'est-à-dire qu'elle ne requiert pas l'inclusion d'informations apportées par une vérité-terrain quelconque.

L'étape qui suit naturellement consiste à séparer les données résultantes afin de regrouper, de manière automatique, les passages qui comportent le même nombre

de personnes : cet objectif est développé dans le chapitre suivant qui fera intervenir en particulier le problème général du partitionnement de données ou *clustering*.

Partitionnement de données

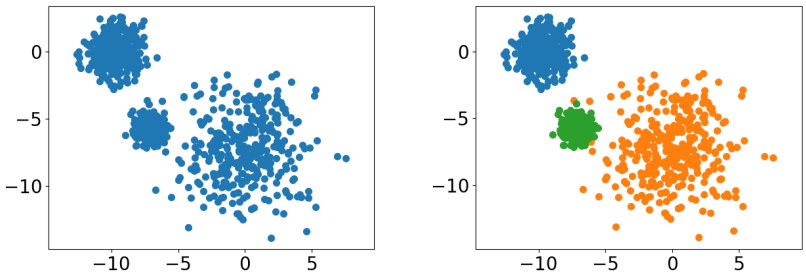
5

Sommaire

5.1 Introduction

Ce chapitre aborde le problème général du partitionnement de données, également appelé *clustering* ou *cluster analysis* en anglais. Les différents termes seront utilisés de manière équivalente dans la suite.

Le clustering est un sous-domaine de l'apprentissage automatique et fait partie des méthodes communes d'exploration et d'analyse de données. Son but primaire est de diviser un jeu de données particulier en sous-groupes, que l'on appelle *clusters*, de telle sorte que les points à l'intérieur d'un même cluster partagent des caractéristiques communes, tout en présentant des dissimilarités avec les données d'un autre groupe. Une illustration simple est donnée en Figure 5.1. A gauche est représentée une base de données en deux dimensions pour laquelle trois clusters se distinguent, la figure de droite montre le résultat d'un algorithme de partitionnement où chaque couleur équivaut à une classe.



(a) Jeu de données original. **(b)** Jeu de données partitionné.

Figure 5.1. Exemple de partitionnement de données.

On peut alors se demander en quoi ce domaine diffère de la classification. La distinction provient principalement du fait qu'ici aucune information concernant l'appartenance de chaque donnée à telle ou telle classe n'est disponible, ni celle du nombre total de classes. On évolue dans un cadre complètement **non-supervisé**. Cela constitue donc une des difficultés majeures liées à ce domaine : le partitionnement de données doit être effectué sans aucune information *a priori*.

Un deuxième challenge réside dans le fait qu'aucune définition exacte de ce qu'est un cluster n'existe [EC02]. Cela a conduit les chercheurs à aborder depuis plusieurs décennies le problème du clustering par différentes approches et à créer de multiples algorithmes tous différents les uns des autres.

Comme dit plus haut, le partitionnement de données présente tout de même l'avantage fondamental qu'aucune information *a priori* sur les données à traiter n'est

nécessaire. En effet, dans la période actuelle, les données abondent grâce à Internet notamment : images, vidéos, signaux, *etc*. Ainsi, obtenir une vérité-terrain, à travers des annotations manuelles bien souvent, est chronophage, coûteux et encombrant. Le domaine du clustering s'abstrait de cette difficulté en tentant d'établir un ordre parmi les données sans information préalable. Cette force fait du partitionnement de données un des pans actifs de recherche.

De nombreuses applications ont pu voir le jour grâce aux progrès dans ce domaine. On peut par exemple citer :

— le regroupement de séquences biologiques liées en bioinformatique [Rem+01],

— la segmentation en marketing [AE01],

— la détection de communauté sur les réseaux sociaux [Dao+20],

— les systèmes de recommandation [SK12],

— la détection d'anomalie [Sya+12],

— *etc*.

Le clustering est particulièrement intéressant pour le projet MIVAO. Nous avons pu voir dans le chapitre précédent qu'une réduction de dimension permettait, à partir des images unitaires, d'extraire des informations précieuses sur chaque passage de véhicule. L'objectif principal est donc d'utiliser le partitionnement de données afin de regrouper les passages pour lesquels le même nombre de passagers sont présents, dans l'optique d'apporter une aide à l'annotation manuelle de milliers de vidéos. En effet, de cette manière, seule une annotation par type de passages est nécessaire, au lieu d'une annotation par passage. A titre d'illustration, si l'on considère les données de la Figure 5.1b comme représentant des images de remontées mécaniques, alors seulement trois annotations manuelles sont nécessaires.

Le reste de ce chapitre se décompose comme suit. Un état de l'art des différents algorithmes de clustering est exposé en section 5.2. Ensuite, nous présentons en sections 5.3 et 5.4 deux méthodes développées durant cette thèse : la première propose une généralisation de la fameuse technique de partitionnement par densité DBSCAN, la deuxième combine la modélisation par mélange Gaussien et le clustering spectral. Précisons que le deuxième algorithme détaillé, contrairement au premier, est encore dans sa forme préliminaire : il sera appliqué dans la suite uniquement à des données synthétiques. Enfin, une conclusion de ce chapitre et des perspectives sont données en section 5.5.

5.2 Etat de l'art

Nous présentons dans cette section les principales approches de partitionnement de données et les algorithmes associés. Le lecteur pourra se référer à [XT15 ; Sax+17] pour des études approfondies. Nous considérons dans la suite un jeu de données $\mathcal{D}$ constitué de n individus $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ parmi lesquelles n_c groupes sont à découvrir. Comme dit plus haut, les approches pour aborder le problème du clustering sont diverses, d'où la catégorisation proposée ci-dessous.

Méthodes par connectivité

Ce premier ensemble de techniques vise à construire une hiérarchie parmi les données basée sur la notion de connectivité, autrement dit de distance. On parle également de **partitionnement hiérarchique**. Pour n_c (le nombre de clusters cachés dans les données) connu, deux manières de procéder existent :

- Ascendante (*agglomerative* ou *bottom-up* en anglais) – Un cluster est initialement créé pour chaque donnée, puis ces derniers sont petit à petit fusionnés jusqu'à obtention de n_c groupes.
- Descendante (*divisive* ou *top-down* en anglais) – L'ensemble des données forme un unique cluster qui est séparé au fur et à mesure jusqu'à convergence vers n_c groupes.

Les deux approches étant complémentaires, nous nous focalisons dans la suite sur les méthodes agglomératives. On peut toutefois citer l'algorithme DIvisive ANAlysis clustering (DIANA) [KR05].

La structure algorithmique est similaire pour l'ensemble des techniques hiérarchiques agglomératives. Elle est donnée en Figure 5.2.

On peut légitimement se demander ce que signifie le terme "dissimilarité". Ce dernier fait intervenir deux concepts. D'abord, une notion de distance entre chaque donnée est nécessaire. Pour deux points $\boldsymbol{x}$ et $\boldsymbol{y}$, différents choix sont imaginables en fonction du contexte de l'application. Il est par exemple possible d'utiliser la distance de Minkowski :

$$\forall p > 0 \quad d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_i |x_i - y_i|^p \right)^{1/p}, \tag{5.1}$$

1. Créer une classe par individu.
2. Tant que le nombre de classes actuel est strictement supérieur à n_c :
   i. Calculer la dissimilarité entre chaque paire de classes.
   ii. Choisir celle qui est minimale.
   iii. Fusionner les deux clusters correspondants.

Figure 5.2. Structure générique d'un algorithme de clustering agglomératif par connectivité. Le nombre n_c de clusters à découvrir est connu à l'avance et est donné en entrée.

à partir de laquelle on peut notamment tirer les distances de Manhattan ($p = 1$), Euclidienne ($p = 2$) et de Chebyshev ($p \to +\infty$), ou bien la distance de Mahalanobis :

$$d_{\mathbf{\Sigma}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \mathbf{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{y})}, \tag{5.2}$$

où $\mathbf{\Sigma}$ est une matrice de covariance, parmi d'autres. Le deuxième point important est l'utilisation d'un critère permettant de mesurer la similarité entre deux groupes de données. Différents critères ont été proposés au fil des années, donnant lieu à plusieurs variantes de l'algorithme. En considérant deux clusters C_1, C_2 et $d(.,.)$ une distance telle que définie précédemment, on peut notamment citer :

— $\min\{d(\boldsymbol{x_1}, \boldsymbol{x_2}) \quad \forall (\boldsymbol{x_1}, \boldsymbol{x_2}) \in C_1 \times C_2\}$, qui donne lieu à la méthode Single-LINKage clustering (SLINK) [Sib73].

— $\max\{d(\boldsymbol{x_1}, \boldsymbol{x_2}) \quad \forall (\boldsymbol{x_1}, \boldsymbol{x_2}) \in C_1 \times C_2\}$, qui aboutit à la variante Complete-LINKage clustering (CLINK) proposée par Defays en 1977 [Def77].

— l'algorithme Unweighted Pair Group Method with Arithmetic mean (UPGMA) [SM58] qui utilise le critère :

$$\frac{1}{|C_1| \times |C_2|} \sum_{\boldsymbol{x_1} \in C_1} \sum_{\boldsymbol{x_2} \in C_2} d(\boldsymbol{x_1}, \boldsymbol{x_2}), \tag{5.3}$$

où $|.|$ représente le cardinal d'un ensemble.

— le critère Ward [War63], proposé en 1963, défini par :

$$\frac{|C_1| \times |C_2|}{|C_1| + |C_2|} d(\boldsymbol{\mu_1}, \boldsymbol{\mu_2}), \tag{5.4}$$

avec $\boldsymbol{\mu_i}$ le barycentre de la classe i.

— *etc.*

Le partitionnement de données par connectivité a déjà été utilisé en pratique, *e.g.* en traitement automatique du langage naturel [Ush96]. Il présente les avantages d'être déterministe et simple à comprendre et à implémenter. Cependant, ce genre de méthodes montre une complexité algorithmique élevée et une difficulté pour interpréter les résultats (sous la forme de dendrogramme), particulièrement quand le nombre de points à partitionner augmente. De plus, le nombre de clusters à découvrir doit être précisé à l'avance ; il peut également être déterminé de manière interactive par l'utilisateur, ce qui n'est pas désirable en pratique.

Méthodes par centroïde

Les algorithmes de ce sous-ensemble cherchent à assigner chaque point au cluster dont le barycentre est le plus proche. Ainsi, on cherche à résoudre le problème d'optimisation suivant :

$$\underset{\boldsymbol{C}}{\operatorname{argmin}} \sum_{i=1}^{n_c} \sum_{\boldsymbol{x_j} \in C_i} \|\boldsymbol{x_j} - \boldsymbol{\mu_i}\|_2^2, \tag{5.5}$$

où $\boldsymbol{C} = \{C_1, \ldots, C_{n_c}\}$.

La méthode la plus célèbre de ce type, voire même de tout le domaine du partitionnement, est K-means [HW79]. Son fonctionnement peut être expliqué comme suit :

1. Choisir des points, au hasard parmi le jeu de données, afin de constituer les moyennes initiales.
2. Affecter chaque donnée au cluster dont la moyenne est la plus proche.
3. Mettre à jour la moyenne de chaque cluster grâce aux nouvelles données assignées.
4. Répéter les deux étapes précédentes jusqu'à convergence.

De nombreuses variantes de cet algorithme existent, telles que **(a)** K-means++ [AV07] qui améliore l'initialisation des moyennes, **(b)** Mini-batch K-means [Scu10] qui permet une application à des bases de données de grande taille, **(c)** Partitioning Around Medoids (PAM) [KR87] qui choisit un point de chaque cluster comme représentant de ce dernier, plutôt qu'une moyenne "fictive" n'existant pas dans le jeu de données original, **(d)** K-medians [Bra+96] qui remplace les moyennes par les médianes, **(e)** Fuzzy C-Means (FCM) [Dun73] qui est la variante de K-means via la logique floue, et **(f)** Kernel K-means [Dhi+04], la version à noyau de K-means.

Une application concrète et classique de K-means est la segmentation des pixels d'une image par couleurs [Dha+15]. Ces algorithmes sont simples et faciles à implémenter mais se heurtent à des difficultés lorsque les clusters ne sont pas sphériques.

Méthodes par distribution

Dans ce cas, les techniques développées approchent le problème d'un point de vue statistique en obtenant une estimation de la distribution des données, via des densités élémentaires. La méthode la plus connue est sûrement celle à base de Mélange Gaussien (MG) : les densités sont alors supposées normales. Par suite, les points appartenant à la même distribution constituent un même cluster. Nous reviendrons sur ces techniques en section 5.4.

Méthodes par densité

Ce genre de techniques définissent un cluster comme une zone de haute densité entourée par un espace de plus faible densité. Nous nous intéressons particulièrement à ces méthodes car elles permettent de déterminer le nombre de clusters de manière automatique : cet élément est crucial dans le cadre du projet MIVAO étant donné que ce nombre, représentant celui de types différents de véhicules présents dans la vidéo, est imprévisible en pratique. On distingue les "types" par le nombre de passagers que chaque véhicule transporte.

L'algorithme précurseur en la matière est Density-Based Spatial Clustering with Applications to Noise (DBSCAN), développé par Ester *et al.* en 1997 [Est+97]. Ces derniers proposent de regrouper les points, parmi le jeu de données $\mathcal{D}$, qui partagent la même densité locale. La méthode dépend de deux paramètres : **(a)** ε, le rayon du voisinage sphérique autour de chaque point P défini par $\mathcal{N}_\varepsilon(\mathrm{P}) = \{\mathrm{Q} \in \mathcal{D} \text{ tel que } \|P-Q\| \leq \varepsilon\}$, et **(b)** $MinPts$, le nombre minimal de points qu'un voisinage doit contenir afin d'être considéré comme valide.

L'algorithme original est donné en pseudo-code en Algorithmes 1 et 2. Il peut être résumé schématiquement (voir [Sch+17]) comme suit :

1. Identifier les données, appelées *core points* en anglais, dont le voisinage $\mathcal{N}_\varepsilon$ contient plus de $MinPts$ éléments.
2. Assembler les core points voisins pour construire des clusters.
3. Assigner chaque non-core point au cluster le plus proche si possible, le considérer comme du bruit sinon.

```
Inputs: 𝒟, ε, MinPts
ClusterId = nextId(NOISE)
for i = 1 to 𝒟.size do
    Point = 𝒟.get(i)
    if Point.ClId = UNDEFINED then
        if ExpandCluster(𝒟, Point, ClusterId, ε, MinPts) then
            ClusterId = nextId(ClusterId)
        end
    end
end
```

Algorithme 1: DBSCAN comme détaillé dans [Est+97].

```
Inputs: 𝒟, Point, ClId, ε, MinPts
seeds = 𝒟.regionQuery(Point, ε)
if seeds.size < MinPts then
    𝒟.changeClId(Point, NOISE)
    return False
end
𝒟.changeClIds(seed, ClId)
seeds.delete(Point)
while seeds is not empty do
    currentP = seeds.first()
    result = 𝒟.regionQuery(currentP, ε)
    if result.size ≥ MinPts then
        for i = 1 to result.size do
            resultP = result.get(i)
            if resultP.ClId in {UNDEFINED, NOISE} then
                if resultP.ClId = UNDEFINED then
                    seeds.append(resultP)
                end
                𝒟.changeClId(resultP, ClId)
            end
        end
    end
    seeds.delete(currentP)
end
return True
```

Algorithme 2: Fonction EXPANDCLUSTER de l'Algorithme 1.

Cette méthode par densité est devenue populaire principalement pour trois raisons :

- Premièrement, le nombre de clusters est déterminé de manière automatique, ce n'est pas un paramètre de DBSCAN. Comme dit plus haut, cet aspect est crucial dans le cadre du projet MIVAO.

- De plus, cet algorithme est capable de détecter les *outliers*, c'est-à-dire les points éloignés de tout cluster et considérés alors comme du bruit.
- Enfin, il ne dépend que de deux paramètres (à savoir ε et $MinPts$) qui peuvent être fixés grâce à des heuristiques [Sch+17].

Bien que DBSCAN présente certains avantages notables, des points faibles sont également à noter : **(a)** la méthode n'est pas entièrement déterministe, c'est-à-dire que différentes exécutions peuvent aboutir à différents résultats, **(b)** elle n'est pas performante quand la densité évolue, que ce soit entre les groupes de données ou à l'intérieur d'un même cluster, et **(c)** le choix des paramètres optimaux nécessite une connaissance approfondie des données.

De nombreuses améliorations ont été proposées. On peut notamment citer :

— GDBSCAN [San+98], une généralisation de DBSCAN à tout genre de données (catégoriques par exemple).

— OPTICS [Ank+99 ; SG18], qui s'abstrait du paramètre ε.

— HDBSCAN [Cam+13], qui combine DBSCAN avec du partitionnement hiérarchique.

— [Tra+13], afin de gérer les cas de clusters denses.

— ADCN [Mai+16 ; Mai+17], proposant une modification pour des données distributées anisotropiquement. Toutefois, cette méthode ne fonctionne que pour des données en deux dimensions.

Nous souhaitons également souligner l'algorithme par densité mean-shift [FH75] : ce dernier vise à déterminer, via estimation de densité par noyau [Par62], les maxima locaux d'une distribution, c'est-à-dire les localisations où les données sont nombreuses. Bien que cette méthode soit capable de découvrir des clusters de n'importe quelle taille, elle est bien souvent complexe à paramétrer.

Méthodes par grille

L'approche utilisée ici consiste à diviser l'espace dans lequel les données évoluent grâce à une grille, constituée de cellules. Les clusters sont formés par des cellules, voisines les unes des autres, dont la densité (*i.e.* le nombre de points présents à l'intérieur d'une cellule) est suffisamment élevée. On retrouve une idée commune avec les méthodes par densité détaillées précédemment. Les deux algorithmes principaux de ce sous-ensemble sont STatistical INformation Grid (STING) [Wan+97] et CLustering In QUEst (CLIQUE) [Agr+98].

Méthodes par corrélation

L'idée derrière ce type de techniques provient de l'observation que des corrélations existent entre les données. Ces dépendances sont locales spatialement : elles peuvent, entre autres, concerner des variables différentes au fur et à mesure que l'on se déplace dans l'espace des données. Trouver ces groupes d'individus corrélés revient à déterminer les différents clusters. Le travail fondateur dans ce domaine est celui de Böhm *et al.* en 2004 qui donne lieu à l'algorithme Computing Clusters of Correlation Connected Objects (4C) [Bö+04]. Notons que ce dernier combine DBSCAN et ACP. Bien que 4C découvre automatiquement les différents groupes de données, quatre paramètres sont à fournir en entrée.

Méthodes par graphe

Le partitionnement est ici réalisé grâce à un graphe pour lequel chaque donnée originale constitue un noeud. Ces techniques se basent sur l'analyse du spectre, *i.e.* des valeurs propres, de la matrice représentative de ce graphe : on parle alors de Clustering Spectral (CS). Les deux algorithmes à la base de ce sous-domaine sont celui de Shi et Malik [SM00], ainsi que celui de Andrew Ng *et al.* en 2001 [Ng+01].

Ce dernier peut être résumé par les étapes suivantes :

1. Créer la matrice d'affinité $\boldsymbol{A} = (a_{ij})$ entre chaque paire de points par :

$$\forall (i,j) \in [\![1,n]\!]^2 \quad a_{ij} = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}{2\sigma^2}\right), \tag{5.6}$$

   où σ est l'écart-type du noyau Gaussien, paramètre de l'algorithme. De plus, les éléments diagonaux de $\boldsymbol{A}$ sont fixés à zéro.
2. Construire la matrice diagonale $\boldsymbol{D}$, somme ligne par ligne de la matrice d'affinité.
3. Obtenir la matrice Laplacienne en normalisant $\boldsymbol{A}$ par $\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$.
4. Décomposer $\boldsymbol{L}$ en éléments propres et récupérer $\boldsymbol{V} \in \mathbb{R}^{n \times n_c}$ les vecteurs propres correspondant aux n_c plus grandes valeurs propres.
5. Normaliser $\boldsymbol{V}$ de telle manière que chaque ligne soit de norme unitaire.
6. Appliquer K-means avec n_c clusters en considérant chaque ligne de $\boldsymbol{V}$ comme une donnée dans $\mathbb{R}^{n_c}$.
7. Assigner l'étiquette de la $i^{\text{ème}}$ ligne de $\boldsymbol{V}$ à la donnée $\boldsymbol{x_i}$ originale.

Des variantes existent afin de déterminer les paramètres σ et n_c optimaux, voir par exemple [ZMP04].

Cette approche spectrale a été utilisée en pratique pour la séparation de la parole via partitionnement de spectrogramme [BJ06]. Les méthodes par graphe permettent de découvrir des clusters de n'importe quelle forme, elles sont toutefois complexes à paramétrer et nécessitent le nombre de clusters en entrée.

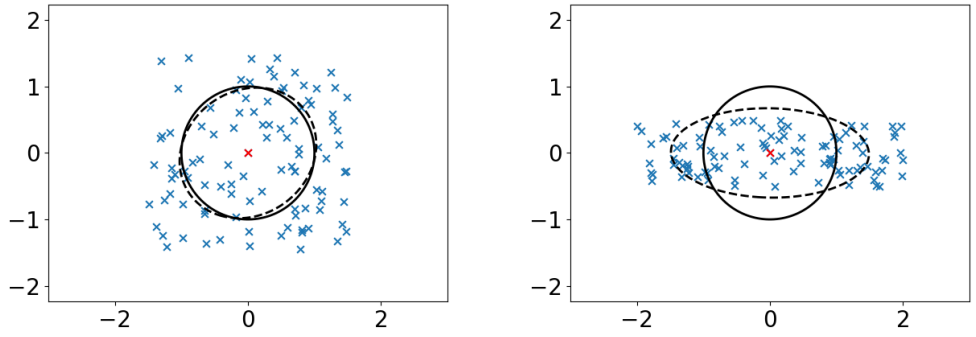
5.3 Ellipsoidal Neighborhood DBSCAN

Nous avons vu dans la section précédente que l'algorithme DBSCAN, bien que puissant, échoue dans certaines configurations, notamment lorsque la densité des données évolue au sein d'un même cluster. La raison principale est l'utilisation de voisinages sphériques identiques en tout point. Pour pallier cette difficulté, nous proposons dans cette section une généralisation de cette méthode, appelée Ellipsoidal Neighborhood Density-Based Spatial Clustering with Applications to Noise (EN-DBSCAN) : en remplaçant les voisinages sphériques par des voisinages ellipsoïdaux, EN-DBSCAN permet une adaptation automatique aux données au niveau local. Une illustration de ce phénomène est donnée en Figure 5.3. L'algorithme proposé montre de multiples caractéristiques intéressantes :

- **Découverte automatique** des différents clusters.
- **Adaptabilité** à n'importe quel problème en dimension d.
- **Faible nombre** de paramètres.
- **Auto-adaptation** à toute distribution, grâce à l'introduction de voisinages ellipsoïdaux.
- **Invariance** à un changement d'échelle de la base de données.

De plus, il sera montré, théoriquement et expérimentalement, que la méthode proposée est effectivement une généralisation de DBSCAN.

Nous souhaitons souligner que l'idée d'utiliser des ellipsoïdes pour le partitionnement de données a déjà été mise en pratique. On peut par exemple citer le travail de Shioda et Tunçel [ST07] dans lequel chaque cluster est couvert par l'ellipsoïde de plus petit volume (via l'estimateur Minimum Volume Ellipsoid [VAR09]), ou bien l'algorithme Gustafson-Kessel [GK78 ; Pat05], utilisant la logique floue et la distance de Mahalanobis [Mah36].



(a) Distribution uniforme. **(b)** Distribution anisotropique.

Figure 5.3. Voisinages sphériques (en traits pleins) et ellipsoïdaux (en pointillés) pour différentes distributions de données. Le voisinage sphérique est identique dans les deux cas alors que celui ellipsoïdal s'adapte automatiquement aux données.

Précisons également que, dans la suite de cette section, par abus de langage, aucune différence ne sera faite entre *sphère* (respectivement *ellipsoïde*) et son homologue en plus grande dimension *hyper-sphère* (respectivement *hyper-ellipsoïde*).

5.3.1 Principales étapes

Etant donné un jeu de données $\mathcal{D} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ composé de n points dans l'espace Euclidien $\mathbb{R}^d$ (d étant le nombre de dimensions), l'algorithme proposé EN-DBSCAN est capable de pratiquer un partitionnement en s'adaptant localement à la distribution des données à travers les voisinages ellipsoïdaux. On peut effectivement noter, à partir de la Figure 5.3, à quel point ces derniers s'adaptent mieux aux données que les sphères communément utilisées.

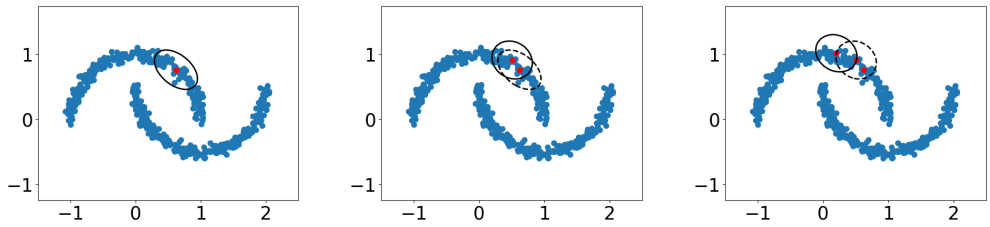
Le modèle proposé est résumé par l'Algorithme 3. Il ne dépend que d'un paramètre, à savoir le nombre minimal $MinPts$ de points qu'un voisinage doit contenir afin d'être considéré comme valide. EN-DBSCAN fonctionne sommairement comme suit : toutes les étiquettes sont initialisées à UNDEFINED. Ensuite, un nouveau cluster est formé à partir d'un point P dont le voisinage contient plus de $MinPts$ éléments, il est sinon considéré comme *outlier* et l'étiquette NOISE lui est assignée (lignes 4 à 12). La seconde étape, de la ligne 13 à la ligne 23, réunit les points similaires à P, voisinage par voisinage : ainsi, ils partagent tous la même étiquette, *i.e.* un entier positif qui correspond à l'indice du cluster auquel ils appartiennent, créé à l'étape précédente. La Figure 5.4 montre les premières itérations de l'algorithme proposé sur un exemple simple.

Inputs: $\mathcal{D}$, $MinPts$
C = 0
Initialize all labels to UNDEFINED
for *each point P in* $\mathcal{D}$ **do**
    if *label(P)* $\neq$ *UNDEFINED* **then**
        continue
    end
    $\mathcal{N}$ = EllipsoidalNeighborhood($\mathcal{D}$, P)
    if $|\mathcal{N}| < MinPts$ **then**
        label(P) = NOISE
        continue
    end
    C = C + 1
    label(P) = C
    $\mathcal{S} = \mathcal{N} \backslash \{P\}$
    while $\mathcal{S} \neq \emptyset$ **do**
        Pick the first element Q in $\mathcal{S}$
        if *label(Q)* = *NOISE* **then**
            label(Q) = C
        end
        if *label(Q)* $\neq$ *UNDEFINED* **then**
            continue
        end
        label(Q) = C
        $\mathcal{N}$ = EllipsoidalNeighborhood($\mathcal{D}$, Q)
        if $|\mathcal{N}| \geq MinPts$ **then**
            $\mathcal{S} = \mathcal{S} \cup \mathcal{N}$
        end
        $\mathcal{S} = \mathcal{S} \backslash \{Q\}$
    end
end

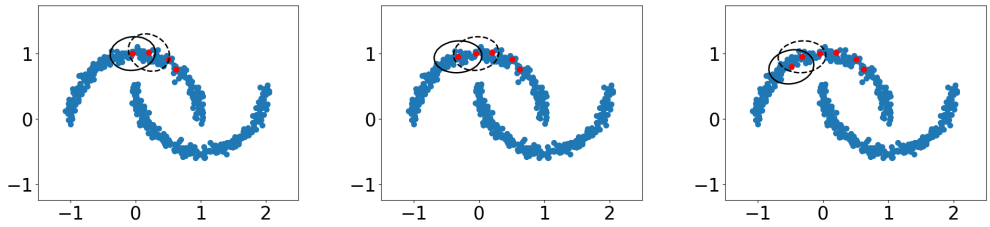
Algorithme 3: EN-DBSCAN en pseudo-code.

La structure de l'algorithme proposé est similaire à celle de DBSCAN : en effet, les contributions apportées résident dans la fonction EllipsoidalNeighborhood. Cette dernière peut se résumer grâce aux étapes de la Figure 5.5.

Ces différentes étapes sont détaillées dans la suite. Comme détaillé en sous-section 5.3.2, les voisinages ellipsoïdaux peuvent être déterminés via la distance de Mahalanobis [Mah36], pour laquelle nous présentons en sous-section 5.3.3 une définition spécifique pour la matrice de covariance impliquée. Les ellipsoïdes calculées montrent des différences en termes de taille, il est donc complexe de tirer une conclusion vis-à-vis du nombre de points inclus dans chacune. Ainsi, nous proposons de les normaliser, comme expliqué en sous-section 5.3.4. Enfin, l'ensemble des éléments



(a) Itération n°1. **(b)** Itération n°2. **(c)** Itération n°3.



(d) Itération n°4. **(e)** Itération n°5. **(f)** Itération n°6.

Figure 5.4. Premières itérations d'EN-DBSCAN sur un exemple simple. L'ellipse courante est représentée en trait plein alors que celle associée au point étiqueté à l'étape précédente (points rouges) est affichée en pointillés.

1. Estimation de la distribution locale autour de chaque point d'intérêt $P \in \mathbb{R}^d$.
2. Détermination de l'ellipsoïde $\mathscr{E}$ associée.
3. Normalisation de l'ellipsoïde $\mathscr{E}$ afin que son volume soit égal à celui d'une sphère de rayon ε.
4. Récupération des points inclus dans l'ellipsoïde normalisée, centrée sur P.

Figure 5.5. Etapes de la fonction EllipsoidalNeighborhood proposée.

inclus dans chaque ellipsoïde doit être récupéré. Cette étape peut être coûteuse étant donné qu'elle requiert le calcul de la distance de Mahalanobis entre chaque donnée et le point d'intérêt. Pour cette raison, la sous-section 5.3.5 détaille une astuce pour accélérer cette étape.

5.3.2 Lien entre ellipsoïde et matrice de covariance

Nous soulignons dans cette sous-section le fait qu'une matrice de covariance quelconque peut être représentée géométriquement par une ellipsoïde, et vice-versa.

Considérons une ellipsoïde $\mathscr{E}$ en d dimensions, centrée sur $\boldsymbol{\mu}_{\mathscr{E}}$, alignée avec les axes canoniques de $\mathbb{R}^d$, dont les demi-axes sont dénotés $a_1, \ldots, a_d$. Un point $\boldsymbol{x}$ reposant sur le contour de $\mathscr{E}$ suit la relation suivante :

$$(\boldsymbol{x} - \boldsymbol{\mu}_{\mathscr{E}})^T \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{\mu}_{\mathscr{E}}) = 1, \tag{5.7}$$

avec

$$\boldsymbol{A} = \begin{pmatrix} 1/a_1^2 & & (0) \\ & \ddots & \\ (0) & & 1/a_d^2 \end{pmatrix}. \tag{5.8}$$

Nous pouvons maintenant introduire le résultat suivant.

Théorème 5.1. *Soit $\mathscr{E}$ une ellipsoïde (en dimension d) centrée sur $\boldsymbol{\mu}_{\mathscr{E}}$, alignée avec les axes canoniques de $\mathbb{R}^d$ et dont les demi-axes sont notés $a_1, a_2, \ldots, a_d$. En échantillonnant de manière **uniforme** des points au niveau de son contour, la matrice de covariance $\boldsymbol{C}_{\mathscr{E}}$ associée est égale à :*

$$\boldsymbol{C}_{\mathscr{E}} = \frac{1}{d} \begin{pmatrix} a_1^2 & & (0) \\ & \ddots & \\ (0) & & a_d^2 \end{pmatrix}, \tag{5.9}$$

où (0) représente une matrice triangulaire de zéros de taille appropriée.

Démonstration. Voir Appendice C. □

Arrêtons-nous un instant pour mettre l'accent sur la notion d'échantillonnage uniforme. En effet, afin de démontrer le théorème précédent, il paraît légitime de choisir pour les points du contour d'une ellipsoïde une paramétrisation classique, par exemple en trois dimensions :

$$\begin{cases} x = a_1 \cos\theta \cos\phi \\ y = a_2 \cos\theta \sin\phi \\ z = a_3 \sin\theta \end{cases} \tag{5.10}$$

avec $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ et $0 \leq \phi < 2\pi$. Une généralisation en n'importe quelle dimension est donnée dans [Blu60]. Cependant, un problème se pose. On peut exprimer un élément de surface $\mathrm{d}\Omega$ avec la formule suivante :

$$\mathrm{d}\Omega = \sin\phi \, \mathrm{d}\theta \, \mathrm{d}\phi. \tag{5.11}$$

On peut observer sur cette dernière équation la dépendance à l'angle ϕ seulement, pas à θ. En pratique, cela signifie que si l'on échantillonne à intervalle régulier les

deux angles précédents, du fait de la non-linéarité des fonctions trigonométriques, alors on obtient un échantillonnage non-uniforme des points à la surface de l'ellipsoïde : la densité sera supérieure au niveau des pôles. Cela introduit par suite des erreurs dans le calcul de la matrice de covariance associée. Cette observation justifie donc le rejet d'une paramétrisation telle que celle de l'Équation 5.10.

En observant la similarité entre l'Équation 5.8 et l'Équation C.4, on obtient un lien clair entre une ellipsoïde et la matrice de covariance associée, à savoir :

$$\boldsymbol{A} = \frac{1}{d}\boldsymbol{C}_{\mathscr{E}}^{-1}. \tag{5.12}$$

Par la suite, le carré de la distance de Mahalanobis [Mah36] entre $\boldsymbol{x}$ et $\boldsymbol{\mu}_{\mathscr{E}}$ peut être calculé, grâce à l'Équation 5.7, par :

$$d_{\mathscr{E}}^2(\boldsymbol{x}, \boldsymbol{\mu}_{\mathscr{E}}) = (\boldsymbol{x} - \boldsymbol{\mu}_{\mathscr{E}})^T \boldsymbol{C}_{\mathscr{E}}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_{\mathscr{E}}) = d. \tag{5.13}$$

Notons que cette distance représente le **critère d'inclusion** d'un point quelconque de $\mathbb{R}^d$ par rapport à l'ellipsoïde $\mathscr{E}$. En effet, si $d_{\mathscr{E}}^2$ est inférieure (respectivement supérieure) à la dimension d, on peut alors en conclure que le point $\boldsymbol{x}$ se situe à l'intérieur (respectivement à l'extérieur) de l'ellipsoïde.

Dans le cas d'ellipsoïdes non-alignées avec les axes canoniques, il est toujours possible de se ramener au cas de figure précédent via rotations.

5.3.3 Matrice de covariance locale

Jusqu'ici, nous avons formalisé le lien entre matrice de covariance et ellipsoïde. Toutefois, l'étape 1 de la fonction EllipsoidalNeighborhood vise à déterminer la distribution locale autour de chaque point d'intérêt. Pour ce faire, nous suggérons d'utiliser la notion de **matrice de covariance locale**, représentée par la matrice de covariance pondérée. Plus précisément, considérons un jeu de données formé par n points $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ en dimension d et un point d'intérêt P. On peut alors calculer la matrice de covariance locale de P comme suit :

$$\boldsymbol{C}_P = \frac{1}{1 - \sum_{i=1}^{n} \tilde{w}_{i,P}^2} \sum_{i=1}^{n} \tilde{w}_{i,P}\ (\boldsymbol{x_i} - \boldsymbol{\mu}^*)^T(\boldsymbol{x_i} - \boldsymbol{\mu}^*), \tag{5.14}$$

où $\tilde{w}_{i,P}$ représente le poids normalisé associé à la donnée $\boldsymbol{x_i}$ par rapport au point P (*i.e.* $\sum_{i=1}^{n} \tilde{w}_{i,P} = 1$) et $\boldsymbol{\mu}^*$ est la moyenne pondérée dont l'expression est :

$$\boldsymbol{\mu}^* = \sum_{i=1}^{n} \tilde{w}_{i,P}\, \boldsymbol{x_i}. \tag{5.15}$$

Dans ce travail, nous avons choisi les différents poids comme étant égaux à l'inverse de la distance Euclidienne quadratique à P, à savoir :

$$w_{i,P} = \begin{cases} 1/\|\boldsymbol{x_i} - P\|_2^2 & \text{si } P \neq \boldsymbol{x_i} \\ (2/\min_{j \neq i} \|\boldsymbol{x_j} - P\|_2)^2 & \text{si } P = \boldsymbol{x_i}. \end{cases} \tag{5.16}$$

Le but de ce choix est de mesurer à quel point la donnée $\boldsymbol{x_i}$ est éloignée du point d'intérêt P : plus elle est proche, plus elle pèse dans le calcul de la matrice de covariance associée à P. La distinction entre les deux cas ci-dessus est faite afin de prendre en compte la non-définition de $w_{P,P}$ à cause d'une division par zéro. Afin de pallier cette difficulté, nous proposons de remplacer la distance Euclidienne $\|P - P\|_2$ par la moitié de celle entre P et son plus proche voisin, d'où le coefficient 2 dans le second cas de l'Équation 5.16. Grâce à cette astuce, $w_{P,P}$ existe et est le plus grand des poids, ce qui assure une présence dominante du point d'intérêt P dans sa propre matrice de covariance locale.

Les poids finaux $\tilde{w}_{i,P}$ sont ensuite obtenus par normalisation :

$$\tilde{w}_{i,P} = \frac{w_{i,P}}{\sum_{j=1}^{n} w_{j,P}} \qquad \forall i \in \{1, \ldots, n\} \tag{5.17}$$

Des exemples de voisinages ellipsoïdaux sont donnés en pointillés en Figure 5.3. On observe que ces derniers décrivent correctement la distribution locale autour du point d'intérêt (représenté par une croix rouge à l'origine sur la figure). De plus, grâce à cette matrice de covariance locale, la méthode s'adapte mieux aux données que si des voisinages sphériques sont utilisés.

Un autre point qu'il est important de souligner est le fait que l'utilisation de matrices de covariance, et donc de la distance de Mahalanobis en Équation 5.13, peut poser un problème en grande dimension : c'est la malédiction de la dimension (*cf.* section 4.1). En effet, une matrice de covariance de taille $d \times d$ comporte $d(d+1)/2$ termes différents, l'évolution du nombre d'éléments est donc quadratique par rapport à la dimension. Ainsi, lorsqu'une telle matrice entre dans le calcul de la distance de Mahalanobis précédente, elle peut engendrer une accumulation de nombreuses

erreurs négligeables et aboutir à une distance finale complètement faussée. Pour contrecarrer ce phénomène, une solution consiste à appliquer une régularisation ℓ_1 aux matrices de covariance locale en suivant l'approche proposée par Friedman *et al.* [Fri+08]. La méthode, nommée *graphical Lasso*, rend la matrice de covariance donnée en entrée parcimonieuse : les éléments hors de la diagonale, si suffisamment petits, sont réduits à zéro. Ainsi, les petites erreurs évoquées précédemment sont réduites et la distance de Mahalanobis résultante prend plus de sens. Nous reviendrons sur ce point ultérieurement.

Afin d'illustrer ces propos, nous considérons 200 points distribués selon la loi normale multivariée de $\mathbb{R}^4$ centrée sur l'origine et dont la matrice de covariance est donnée en Figure 5.6a [1]. En Figure 5.6b est présentée la matrice de covariance empirique déterminée à partir des données échantillonnées. La Figure 5.6c montre la matrice de covariance obtenue après régularisation par l'algorithme *graphical Lasso* : on peut noter que les valeurs sur la diagonale sont peu modifiées alors que celles non-diagonales tendent vers zéro, voire sont égales à zéro (deuxième ligne/colonne par exemple). Ainsi, cette matrice régularisée se rapproche de la matrice de covariance réelle.

$$\begin{pmatrix} 0,8 & 0 & 0,2 & 0,1 \\ 0 & 0,4 & 0 & 0 \\ 0,2 & 0 & 0,3 & 0 \\ 0,1 & 0 & 0 & 0,7 \end{pmatrix}$$

(a) Originale.

$$\begin{pmatrix} 0,837 & 0,016 & 0,225 & 0,073 \\ 0,016 & 0,372 & -0,001 & -0,034 \\ 0,225 & -0,001 & 0,286 & -0,002 \\ 0,073 & -0,034 & -0,002 & 0,715 \end{pmatrix}$$

(b) Empirique.

$$\begin{pmatrix} 0,832 & 0 & 0,188 & 0,036 \\ 0 & 0,37 & 0 & 0 \\ 0,188 & 0 & 0,285 & 0,008 \\ 0,036 & 0 & 0,008 & 0,712 \end{pmatrix}$$

(c) Régularisée.

Figure 5.6. Matrices de covariance originale, empirique et régularisée via l'algorithme *graphical Lasso* [Fri+08].

Enfin, il peut arriver en pratique que la matrice de covariance locale ne soit pas correctement définie. Par exemple, lorsque sont déterminant est nul, le processus d'inversion n'est pas réalisable, il en est de même pour les traitements postérieurs. Cette situation apparaît notamment quand des données voisines présentent la même valeur pour une variable particulière, ce qui aboutit à une variance nulle. Nous proposons d'anticiper cette complication comme suit :

1. Cet exemple est directement inspiré de celui présenté sur le site de la bibliothèque Scikit-learn.

1. La matrice de covariance mal définie $\boldsymbol{C}$ est d'abord décomposée en éléments propres. On obtient alors $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, où $\boldsymbol{V}$ représente les vecteurs propres et $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_d)$ est la matrice diagonale des valeurs propres.

2. On obtient ensuite $\tilde{\boldsymbol{\Lambda}}$ en remplaçant les valeurs propres négatives ou nulles de $\boldsymbol{\Lambda}$ par $\min\{\lambda_i | \lambda_i > 0\}/10$.

3. Enfin, la matrice de covariance corrigée est déterminée par $\tilde{\boldsymbol{C}} = \boldsymbol{V}\tilde{\boldsymbol{\Lambda}}\boldsymbol{V}^{-1}$. Cette dernière est véritablement une matrice semi-définie positive et peut légitimement être utilisée pour des traitements ultérieurs.

5.3.4 Normalisation du volume d'une ellipsoïde

Les matrices de covariances locales, telles que définies dans la sous-section précédente, donnent une idée de la distribution locale des données autour de chaque point d'intérêt. Toutefois, ces dernières dépendant fortement du voisinage proche et surtout de la distance à ces voisins, elles peuvent aboutir à des ellipsoïdes dont les tailles sont différentes. De ce fait, il est difficile de "comparer" de telles ellipsoïdes : par exemple, que peut-on dire d'un premier voisinage incluant cinq données et d'un deuxième voisinage, contenant lui aussi cinq points, mais dont le volume est six fois supérieur au premier ?

Nous présentons ici une normalisation des ellipsoïdes déterminées, correspondant à l'étape 3 de la fonction EllipsoidalNeighborhood que nous proposons. L'idée sous-jacente consiste à modifier la taille de chaque ellipsoïde, agrandir ou rétrécir tout en gardant la même orientation, de telle manière que son volume soit égal à celui d'une sphère de rayon ε.

Considérons le cas général d'une ellipsoïde définie par son centre $\boldsymbol{\mu}$, sa matrice de covariance $\boldsymbol{\Sigma}$ et un scalaire positif r. Alors, n'importe quelle donnée $\boldsymbol{x} \in \mathbb{R}^d$ reposant sur son contour est régie par l'équation suivante :

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = r^2. \tag{5.18}$$

Précisons que le scalaire r joue le même rôle que le rayon d'une sphère. Plus (respectivement moins) il est élevé, plus (respectivement moins) l'ellipsoïde correspondante est volumineuse. L'équation précédente peut être réécrite sous la forme

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \left(r^2 \boldsymbol{\Sigma}\right)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = 1. \tag{5.19}$$

On peut alors déterminer le volume V de cette ellipsoïde par :

$$V = V_d|r^2\mathbf{\Sigma}|^{1/2} = V_d|\mathbf{\Sigma}|^{1/2}r^d, \tag{5.20}$$

où V_d est le volume de l'hyper-sphère unitaire en dimension d et $|\mathbf{\Sigma}|$ est le déterminant de la matrice de covariance. Nous souhaitons que ce volume soit égal à celui d'une sphère de rayon ε, ce qui donne :

$$V_d|\mathbf{\Sigma}|^{1/2}r^d = V_d\varepsilon^d \Leftrightarrow r = \frac{\varepsilon}{|\mathbf{\Sigma}|^{1/(2d)}}. \tag{5.21}$$

En injectant cette expression de r dans l'Équation 5.18, on obtient une formulation simple du critère d'inclusion de n'importe quel point dans l'ellipsoïde dont le volume est normalisé.

Le rayon ε mentionné précédemment constitue ainsi un second paramètre, avec $MinPts$, de la méthode proposée : ce dernier remplit le même rôle que l'argument du même nom dans l'algorithme DBSCAN.

5.3.5 Accélération algorithmique sur le critère d'inclusion

L'étape 4 de la méthode proposée vise à récupérer l'ensemble des points inclus dans chaque ellipsoïde. Elle requiert le calcul du critère d'inclusion, à travers une distance de Mahalanobis similaire à celle de l'Équation 5.13, pour chaque point du jeu de données. On peut à juste titre comprendre à quel point cette étape est onéreuse, notamment lorsque la dimensionalité du problème et/ou le nombre de données augmentent. Nous proposons ainsi une astuce afin d'en réduire la charge computationnelle.

L'idée générale est la suivante : certaines données sont tellement éloignées du point d'intérêt P qu'il est impossible qu'elles se trouvent à l'intérieur de l'ellipsoïde générée par P. Le calcul de la distance de Mahalanobis dans ce cas-là est donc inutile. A titre d'illustration, considérons l'ellipse représentée en Figure 5.7 en trait plein noir, dont les demi-axes sont égaux à a_1 et a_2. On peut alors construire le rectangle circonscrit à cette ellipse, puis le cercle circonscrit à cette boîte englobante. Géométriquement, il est clair que le rayon du cercle est égal à $\sqrt{a_1^2 + a_2^2}$ d'après le théorème de Pythagore. De plus, n'importe quelle donnée située à l'extérieur de ce cercle circonscrit repose nécessairement à l'extérieur de l'ellipse. Dernier élément mais pas des moindres, une simple distance Euclidienne est requise pour savoir si un point est situé à l'intérieur

ou à l'extérieur du cercle circonscrit : ce procédé permet donc de ne calculer une distance de Mahalanobis, coûteuse, uniquement quand cela est nécessaire.

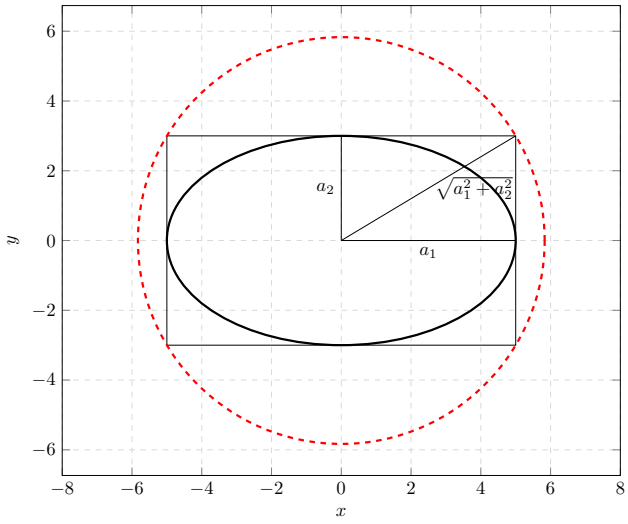


Figure 5.7. Ellipse de demi-axes a_1 et a_2 (en trait plein noir), boîte englobante et cercle circonscrit (en pointillés rouges).

Théorème 5.2. *Soit $\mathscr{E}$ une ellipsoïde en dimension d, centrée sur l'origine, alignée avec les axes canoniques, dont les demi-axes sont $a_1, \ldots, a_d$ et $\boldsymbol{x}$ un point de $\mathbb{R}^d$. Si $\boldsymbol{x}$ se situe en dehors de la sphère centrée sur l'origine et de rayon $\sqrt{a_1^2 + \cdots + a_d^2}$, alors $\boldsymbol{x}$ est également à l'extérieur de l'ellipsoïde $\mathscr{E}$.*

Démonstration. Nous allons démontrer la contraposée. Supposons donc que $\boldsymbol{x}$ appartienne à l'ellipsoïde. On peut alors en déduire $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \leq 1$ avec

$$\boldsymbol{A} = \begin{pmatrix} 1/a_1^2 & & (0) \\ & \ddots & \\ (0) & & 1/a_d^2 \end{pmatrix}. \tag{5.22}$$

Créons la matrice $\boldsymbol{B} = 1/(a_1^2 + \cdots + a_d^2) \times \boldsymbol{I_d}$, où $\boldsymbol{I_d}$ est la matrice identité en dimension d. Etant donné que

$$\forall i \in \{1, \ldots, d\} \quad 0 < \frac{1}{a_1^2 + \cdots + a_d^2} \leq \frac{1}{a_i^2} \tag{5.23}$$

et que les formes suivantes sont quadratiques, on en déduit

$$\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} \leq \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \leq 1. \tag{5.24}$$

On tire de cette dernière équation

$$\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = 1/(a_1^2 + \cdots + a_d^2) \boldsymbol{x}^T \boldsymbol{x} \leq 1 \tag{5.25}$$

puis

$$\boldsymbol{x}^T \boldsymbol{x} = \|\boldsymbol{x}\|_2^2 \leq a_1^2 + \cdots + a_d^2. \tag{5.26}$$

Cette inégalité stipule que le carré de la norme Euclidienne de $\boldsymbol{x}$ est inférieur ou égal à $r_s^2 = a_1^2 + \cdots + a_d^2$: $\boldsymbol{x}$ appartient ainsi à la sphère centrée sur l'origine et de rayon r_s. Notons que ce dernier peut également s'exprimer comme $\sqrt{\mathrm{Tr}\left(\boldsymbol{A}^{-1}\right)}$, où $\mathrm{Tr}(.)$ représente la trace d'une matrice.

La démonstration dans le cas d'une ellipsoïde non centrée sur l'origine est similaire. □

Le résultat qui précède est satisfaisant mais ne nous intéresse pas directement dans le cadre de l'algorithme EN-DBSCAN. En effet, les ellipsoïdes auxquelles nous avons affaire sont régies par l'équation suivante, après normalisation telle qu'expliquée en sous-section 5.3.4 :

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = \left(\frac{\varepsilon}{|\boldsymbol{\Sigma}|^{1/(2d)}} \right)^2. \tag{5.27}$$

En divisant cette équation par le terme de droite, on se retrouve dans la situation du Théorème 5.2 avec

$$\boldsymbol{A} = \left(\frac{|\boldsymbol{\Sigma}|^{1/(2d)}}{\varepsilon} \right)^2 \boldsymbol{\Sigma}^{-1}. \tag{5.28}$$

Par suite, le rayon r_s de la sphère au-delà duquel il est certain que la donnée $\boldsymbol{x}$ n'appartienne pas à l'ellipsoïde peut s'exprimer ainsi :

$$\begin{aligned} r_s &= \sqrt{\mathrm{Tr}\left(\boldsymbol{A}^{-1}\right)} && (5.29) \\ &= \sqrt{\mathrm{Tr}\left(\left(\frac{\varepsilon}{|\boldsymbol{\Sigma}|^{1/(2d)}} \right)^2 \boldsymbol{\Sigma} \right)} && (5.30) \\ &= \sqrt{\left(\frac{\varepsilon}{|\boldsymbol{\Sigma}|^{1/(2d)}} \right)^2 \mathrm{Tr}\left(\boldsymbol{\Sigma}\right)} && (5.31) \\ &= \frac{\varepsilon}{|\boldsymbol{\Sigma}|^{1/(2d)}} \sqrt{\mathrm{Tr}\left(\boldsymbol{\Sigma}\right)} && (5.32) \end{aligned}$$

Faisons une digression afin d'expliquer pourquoi les résultats ci-dessus sont inchangés pour des ellipsoïdes non-alignées avec les axes de $\mathbb{R}^d$. Dans ce cas, on peut calculer

la matrice de covariance associée $\boldsymbol{\Sigma_R}$ et il existe une matrice de rotation $\boldsymbol{R}$, de taille $d \times d$, telle que

$$\boldsymbol{\Sigma_R} = \boldsymbol{R}\ \boldsymbol{\Sigma}\ \boldsymbol{R}^T, \tag{5.33}$$

avec $\boldsymbol{\Sigma}$ la matrice de covariance de l'ellipsoïde alignée (similaire à celle de l'Équation C.4). Ensuite, on peut obtenir le déterminant et la trace de cette matrice comme suit :

$$|\boldsymbol{\Sigma_R}| = |\boldsymbol{R}|.|\boldsymbol{\Sigma}|.|\boldsymbol{R}^T| = 1 \times |\boldsymbol{\Sigma}| \times 1 = |\boldsymbol{\Sigma}| \tag{5.34}$$

$$\mathrm{Tr}\left(\boldsymbol{\Sigma_R}\right) = \mathrm{Tr}\left(\boldsymbol{R\Sigma R}^T\right) = \mathrm{Tr}\left(\boldsymbol{R}^T\boldsymbol{R\Sigma}\right) = \mathrm{Tr}\left(\boldsymbol{\Sigma}\right) \tag{5.35}$$

Comme on peut le voir, le déterminant et la trace d'une matrice de covariance sont invariants aux rotations, il en est de même pour le rayon r_s obtenu à l'Équation 5.32.

Cette méthode présente l'avantage majeur de ne pas nécessiter la connaissance complète de la structure de l'ellipsoïde (*i.e.* ses demi-axes et son orientation) : seule la matrice de covariance, qui d'ailleurs englobe toutes ces informations, est requise.

5.3.6 Complexité algorithmique

Après avoir détaillé les différentes étapes de l'algorithme EN-DBSCAN, penchons-nous un instant sur sa complexité computationnelle. Pour ce faire, considérons sa scalabilité vis-à-vis de deux critères, à savoir comment son temps d'exécution évolue par rapport au nombre et à la dimension des données à partitionner. Dans la suite, nous supposons un jeu de données composé de n individus en dimension d.

La complexité dans le pire des cas de l'algorithme original DBSCAN est $O(n^2)$, étant donné qu'une recherche de voisinage est requise pour chacun des points du jeu de données. Cependant, si une structure d'indexage efficace est utilisée, la complexité est réduite à $O(n \log n)$ [Est+97].

Quant à la technique proposée, la fonction de recherche de voisinage est remplacée par celle présentée en sous-section 5.3.1. Cette dernière consiste dans un premier temps à déterminer une matrice de covariance locale qui peut être fait en $O(d^2 n)$. L'inverse de cette matrice doit ensuite être calculée, tout comme son déterminant (afin de normaliser l'ellipsoïde correspondante) : chacune de ces étapes requiert une complexité temporelle de $O(d^3)$. Enfin, la recherche de voisinage est obtenue via une mesure de formes quadratiques (voir Équation 5.13), qui équivaut à $O(d^2 n)$. En

combinant tous ces éléments et en gardant à l'esprit que ce processus est exécuté pour chaque donnée, nous obtenons une complexité finale en temps égale à $O(d^2n^2 + d^3n)$.

De plus, l'utilisation d'une structure d'indexage efficace et l'astuce expliquée en sous-section 5.3.5 permet de réduire la complexité à $O(d^2 n \log n + d^3 n)$.

5.3.7 Réduction à DBSCAN

Cette sous-section a pour objectif de montrer en quoi la méthode proposée EN-DBSCAN est une généralisation de l'algorithme sur lequel elle s'appuie, à savoir DBSCAN. Pour ce faire, considérons une ellipsoïde $\mathscr{E}$ de centre $\boldsymbol{\mu}$ et de matrice de covariance $\boldsymbol{\Sigma}$. Ainsi, n'importe quel point $\boldsymbol{x}$ situé sur son contour, après normalisation, répond à l'équation :

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = \frac{\varepsilon^2}{|\boldsymbol{\Sigma}|^{1/d}}. \tag{5.36}$$

Supposons de plus que la matrice de covariance $\boldsymbol{\Sigma}$ soit diagonale et dont les éléments non nuls soient égaux à ε^2, autrement dit :

$$\boldsymbol{\Sigma} = \varepsilon^2 \boldsymbol{I_d}. \tag{5.37}$$

On peut ensuite calculer son inverse et son déterminant comme suit :

$$\boldsymbol{\Sigma}^{-1} = \frac{1}{\varepsilon^2} \boldsymbol{I_d} \tag{5.38}$$

$$|\boldsymbol{\Sigma}| = \varepsilon^{2d} \tag{5.39}$$

Les éléments précédents peuvent donc être remplacés dans l'Équation 5.36, on obtient alors :

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = \frac{\varepsilon^2}{|\boldsymbol{\Sigma}|^{1/d}} \tag{5.40}$$

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \frac{1}{\varepsilon^2} \boldsymbol{I_d} (\boldsymbol{x} - \boldsymbol{\mu}) = \frac{\varepsilon^2}{\left(\varepsilon^{2d}\right)^{1/d}} \tag{5.41}$$

$$\frac{1}{\varepsilon^2} (\boldsymbol{x} - \boldsymbol{\mu})^T (\boldsymbol{x} - \boldsymbol{\mu}) = \frac{\varepsilon^2}{\varepsilon^2} \tag{5.42}$$

$$\|\boldsymbol{x} - \boldsymbol{\mu}\|_2^2 = \varepsilon^2 \tag{5.43}$$

$$\|\boldsymbol{x} - \boldsymbol{\mu}\|_2 = \varepsilon \tag{5.44}$$

Nous pouvons conclure de cette égalité que la donnée $\boldsymbol{x}$ appartient à la sphère centrée sur $\boldsymbol{\mu}$ et de rayon ε. Ainsi, nous venons de montrer que EN-DBSCAN est une généralisation de DBSCAN, car le même fonctionnement est obtenu lorsque la matrice de covariance est contrainte à être proportionnelle à la matrice identité.

5.3.8 Résultats expérimentaux

Nous présentons ici les expériences menées et les résultats obtenus par notre méthode EN-DBSCAN ainsi que trois autres algorithmes de partitionnement de données par densité, à savoir OPTICS, DBSCAN et HDBSCAN.

Jeux de données et métrique

Afin d'évaluer les différents algorithmes de partitionnement, nous utilisons les données du CLUSTERING BENCHMARK [2]. Ce dernier est constitué de données synthétiques, mais également issues de situations réelles.

Les partitionnements prédits sont comparés à la vérité-terrain via le score de Fowlkes-Mallows [FM83], défini comme suit :

$$FM = \sqrt{\frac{VP}{VP+FP} \times \frac{VP}{VP+FN}}. \tag{5.45}$$

Le score de Fowlkes-Mallows est donc le pendant de la F-mesure en classification. Il approche 1 dans le cas d'un clustering prédit proche de la vérité-terrain et tend vers 0 si ce dernier est aléatoire.

Précisons toutefois que les définitions de VP, VN, FP et FN sont légèrement modifiées dans le cadre du partitionnement de données. En effet, l'étiquette de chaque donnée, c'est-à-dire l'indice du cluster auquel cette dernière est associée, n'a aucune signification particulière. *A contrario*, en classification, dans le cas d'un filtre anti-spam par exemple, les données que sont les e-mails sont classifiées comme normaux (0) ou spams (1) : les étiquettes représentent ici les différentes classes. Considérons deux points $\boldsymbol{x}$ et $\boldsymbol{y}$ d'un même jeu de données et $\mathscr{P}_1, \mathscr{P}_2$ deux partitionnements : ces derniers peuvent être issus de deux algorithmes différents ou bien d'un seul et de la vérité-terrain. Quatre cas se présentent alors à nous :

2. https://github.com/deric/clustering-benchmark

1. Si $\boldsymbol{x}$ et $\boldsymbol{y}$ appartiennent au même cluster dans $\mathscr{P}_1$ ainsi que dans $\mathscr{P}_2$, alors le compteur VP est incrémenté d'une unité.
2. Si $\boldsymbol{x}$ et $\boldsymbol{y}$ appartiennent au même cluster dans $\mathscr{P}_1$ mais pas dans $\mathscr{P}_2$, alors le compteur FP est incrémenté d'une unité.
3. Si $\boldsymbol{x}$ et $\boldsymbol{y}$ appartiennent au même cluster dans $\mathscr{P}_2$ mais pas dans $\mathscr{P}_1$, alors le compteur FN est incrémenté d'une unité.
4. Si $\boldsymbol{x}$ et $\boldsymbol{y}$ n'appartiennent pas au même cluster, ni dans $\mathscr{P}_1$, ni dans $\mathscr{P}_2$, alors le compteur VN est incrémenté d'une unité.

La comparaison se fait ainsi pour chaque paire de données, d'où :

$$\text{VP} + \text{FP} + \text{FN} + \text{VN} = \frac{n(n-1)}{2}. \tag{5.46}$$

Toutefois, les méthodes par densité sont capables de détecter les *outliers* grâce à l'étiquette NOISE. Afin de s'assurer que le score final ne considère pas ces points comme faisant partie d'un vrai cluster, nous déterminons d'abord le pourcentage d'aire couverte par :

$$\%\_\text{couvert} = 1 - \frac{\#\text{ points étiquetés NOISE}}{\#\text{ points}}. \tag{5.47}$$

Nous reportons dans la suite le score Weighted Fowlkes-Mallows (WFM), c'est-à-dire le score FM obtenu pondéré par le pourcentage d'aire couverte.

Etape courante dans les applications réelles en apprentissage automatique, nous réalisons une normalisation des données, à savoir centrage et réduction par rapport à la moyenne et à l'écart-type de chaque caractéristique, avant d'exécuter les différents algorithmes.

Dans la suite, les paramètres sont fixés comme suit :

- $MinPts$ est choisi comme étant égal au double de la dimensionnalité du jeu de données considéré, comme suggéré dans [San+98].
- ε est cherché dans une plage de 20 valeurs comprises entre 0 et 1, due à la normalisation des données.

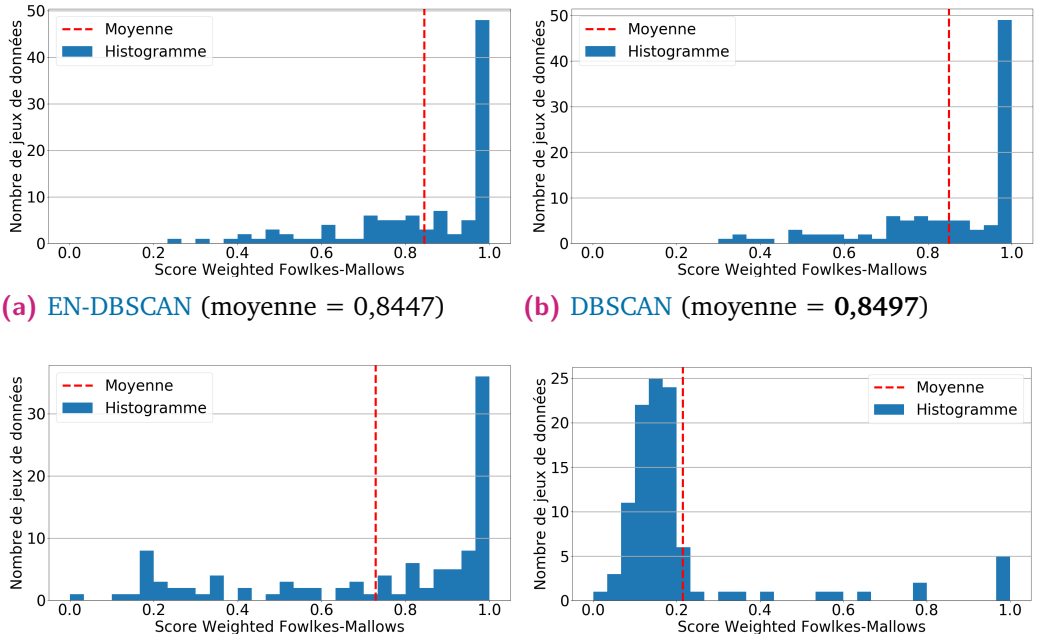
Pour chaque jeu de données et chaque méthode, une recherche exhaustive des paramètres optimaux, *i.e.* ceux qui aboutissent au score WFM le plus élevé, est effectuée.

De plus, nous avons pu noter durant nos expérimentations que l'utilisation de matrices de covariance diagonales, avec les éléments diagonaux différents les uns

des autres, engendrait les meilleures performances. Cette remarque fait écho à la discussion sur la malédiction de la dimension en sous-section 5.3.2. On peut expliquer cette observation par le fait que les covariances entre variables (*i.e.* les éléments non diagonaux) faibles influencent la distance de Mahalanobis dans la mauvaise direction et amène à une “classification” erronée de certains points, par exemple l'inclusion dans une ellipsoïde d'une donnée appartenant à un autre cluster. D'un point de vue géométrique, cela signifie que les ellipsoïdes correspondantes sont alignées avec les axes canoniques, aucune rotation n'est considérée. Dans la suite de cette sous-section, les résultats de EN-DBSCAN sont issus de cette variante.

Données synthétiques

Nous nous attardons ici sur des jeux de données artificielles, en deux ou trois dimensions. Les résultats des quatre algorithmes sont affichés en Figure 5.8 sous la forme d'histogrammes (calculés sur 30 *bins* entre 0 et 1) des scores WFM sur l'ensemble des jeux de données. La moyenne globale est représentée par une ligne verticale rouge en pointillés.



(a) EN-DBSCAN (moyenne = 0,8447) **(b)** DBSCAN (moyenne = **0,8497**)

(c) HDBSCAN (moyenne = 0,7286) **(d)** OPTICS (moyenne = 0,2150)

Figure 5.8. Histogrammes des scores de Fowlkes-Mallows obtenus par quatre méthodes sur des données artificielles.

Bien que EN-DBSCAN présente une diminution en ce qui concerne la moyenne des scores WFM sur l'ensemble des bases de données en comparaison de DBSCAN, elle

est négligeable : seulement 0,5% de différence en moyenne. Pour le prouver, nous réalisons un test statistique de Kolmogorov-Smirnov à deux échantillons [Smi39], qui vérifie l'adéquation entre deux distributions de probabilité. Selon ce dernier, nous ne pouvons pas rejeter l'hypothèse nulle, selon laquelle les distributions de DBSCAN et d'EN-DBSCAN sont identiques, avec une valeur p égale à 0,999. Nous pouvons expliquer ce résultat par le fait que les données artificielles, créées manuellement, ne représentent pas tous les types de situations, surtout les cas où les voisinages ellipsoïdaux dominent par rapport à ceux sphériques. Nous pouvons également noter qu'EN-DBSCAN surclasse HDBSCAN et OPTICS.

Finalement, des expérimentations pour confirmer le fait que l'algorithme proposé est effectivement une généralisation de DBSCAN ont été menées. Nous avons contraint EN-DBSCAN à utiliser la même matrice de covariance pour chacune des données, proportionnelle à la matrice identité (voir sous-section 5.3.7 pour plus de détails). Il s'avère que les résultats obtenus sont exactement égaux. De fait, nous ne les présentons pas ici car ils n'apportent aucune information intéressante.

Données réelles

Nous considérons maintenant sept jeux de données réelles issus du CLUSTERING BENCHMARK. En médiane, chaque jeu comporte 215 points et 7 variables.

Jeu de données	EN-DBSCAN	DBSCAN	HDBSCAN	OPTICS
BALANCE-SCALE	**0,6557**	**0,6557**	0,6089	0,4305
ECOLI	**0,5453**	0,4592	0,5294	0,5197
GLASS	0,2387	0,1498	**0,4529**	0,0568
HABERMAN	0,7267	**0,7377**	0,1627	0,1496
IRIS	0,7409	0,7409	**0,7533**	0,1737
TAE	**0,3120**	0,2237	0,1708	0,1625
THY	0,6859	0,6814	**0,7122**	0,6221
Moyenne	**0,5579**	0,5212	0,4843	0,3021

Table 5.1. Scores WFM obtenus par EN-DBSCAN et trois autres méthodes sur sept jeux de données réelle.

On peut observer en Tableau 5.1 à quel point notre méthode surclasse en moyenne les trois autres algorithmes de clustering. Plus précisément, EN-DBSCAN obtient

le meilleur score WFM pour trois jeux de données et le second meilleur dans trois autres cas, seulement quelques pourcentagers derrière le meilleur score. Ces résultats montrent que prendre en compte la distribution locale de chaque point aboutit à un meilleur partitionnement.

Toutefois, on peut observer que EN-DBSCAN échoue pour le jeu de données GLASS. En réalité, en analysant les données de plus près, on peut se rendre compte que certaines caractéristiques présentent une faible variabilité, ce qui rend difficile le calcul de voisinages ellipsoïdaux, comme expliqué en sous-section 5.3.3.

5.3.9 Conclusion

Nous avons proposé dans cette section l'algorithme EN-DBSCAN, une méthode de partitionnement de données par densité qui utilise des voisinages ellipsoïdaux au lieu de sphères comme c'est le cas dans DBSCAN. De cette manière, l'algorithme permet une auto-adaptation à la distribution locale de n'importe quel jeu de données. Nous avons formalisé une définition de l'ellipsoïde basée sur la matrice de covariance pondérée, ainsi que sa normalisation afin de permettre une comparaison aisée. La distance de Mahalanobis est utilisée pour déterminer si une donnée appartient à tel ou tel voisinage. Cette dernière étant particulièrement onéreuse à calculer pour tous les points de la base de données, nous avons également détaillé une astuce permettant d'accélérer le temps d'exécution de cette étape. De plus, il a été démontré, théoriquement et expérimentalement, que la méthode proposée était une généralisation de DBSCAN. Enfin, les résultats, sur des données synthétiques et réelles, montrent la surperformance de EN-DBSCAN par rapport à d'autres algorithmes de partitionnement, tels que DBSCAN, HDBSCAN et OPTICS. Ce travail a été soumis pour révision à une conférence internationale et est en attente d'acceptation :

> J. Muzeau, M. Oliver-Parera, P. Ladret et P. Bertolino, "EN-DBSCAN : Density-Based Clustering through Adaptive Ellipsoidal Neighborhood", Winter Conference on Applications of Computer Vision, 2021 (en attente d'acceptation).

Plusieurs perspectives s'offrent à nous pour améliorer ce travail. D'abord, il serait intéressant de déterminer une estimation non-biaisée de la matrice de covariance pondérée, afin d'obtenir une représentation géométrique (via l'ellipsoïde) plus précise de la distribution locale. Ensuite, un autre choix pour les poids $w_{i,P}$ de la matrice de covariance locale est possible, plutôt que simplement l'inverse des distances Euclidiennes quadratiques. De plus, il semble que l'inclusion de toutes

les données dans le calcul de ces poids ne soit pas nécessaire, étant donné que les points les plus éloignés ne pèsent pas spécialement : choisir uniquement 95% (par exemple) des voisins pourrait permettre une accélération logicielle avec une perte d'information minimale. Enfin, nous espérons pouvoir inclure le concept de voisinage ellipsoïdal présenté dans ce travail dans la procédure HDBSCAN.

5.4 Gaussian Spectral Clustering

5.4.1 Introduction

Nous nous concentrons dans cette section sur les algorithmes de **partitionnement de données par distribution** qui traitent le problème du clustering d'un point de vue statistique en considérant la densité de probabilité des données. En particulier, nous étudions le cas du clustering par Mélange Gaussien (MG) qui vise à modéliser les données via une combinaison de distributions normales : chaque Gaussienne représente ainsi un regroupement de points.

La détermination du nombre optimal de composantes d'un MG à partir des données uniquement est encore un domaine actif de recherche. Trois stratégies existent pour traiter cette question. La première consiste à initialiser le Modèle de Mélange Gaussien (MMG) avec un nombre faible de composantes, puis à l'augmenter petit à petit jusqu'à convergence [Ued+00 ; Zha+03 ; LL09]. La deuxième approche, opposée à la précédente, vise à fusionner itérativement certaines composantes jusqu'à ce qu'une condition d'arrêt soit rencontrée [FJ02]. Enfin, le dernier groupe de méthodes applique un processus d'optimisation afin de minimiser un critère sur l'ensemble des nombres de composantes possibles [MR14]. Différents critères ont été proposés tels que le Bayesian Information Criterion (BIC) [Sch78], le Minimum Message Length (MML) [Wal05] ou encore le Akaike Information Criterion (AIC) [Aka74].

Toutefois, les approches mentionnées précédemment peuvent souffrir d'un problème récurrent en apprentissage automatique. En effet, dans ces cas-là, la sélection du modèle optimal est basée sur la minimisation de la fonction de vraisemblance $\mathcal{L}$. Cependant, l'ajout de composantes entraîne une diminution de $\mathcal{L}$. Ainsi, le MMG résultant contient un trop grand nombre de composantes : le modèle représente précisément la densité des données mais surestime le nombre de véritables clusters. On parle alors de **surapprentissage**. De plus, le partitionnement de données par MG échoue dans le cas de clusters de forme autre qu'elliptique.

Afin de pallier ces difficultés et de déterminer **automatiquement** le nombre correct de clusters, peu importe leur distribution, nous proposons un algorithme, nommé Gaussian Spectral Clustering (GSC), non-paramétrique, qui fonctionne en trois temps.

1. Premièrement, les données sont modélisées par un MMG choisi de manière optimale grâce à la minimisation du BIC. Comme expliqué précédemment, le modèle résultant est victime de surapprentissage : il est composé de bien plus de composantes que le nombre de groupes présents dans les données.
2. Certaines composantes du modèle précédent, du fait de leur grand nombre, exhibent un fort recouvrement les unes avec les autres. Ce cas de figure signifie souvent que deux composantes, au moins, appartiennent en réalité au même cluster. Ainsi, afin de décider si tel est le cas, une estimation de la similarité entre Gaussiennes est obtenue via le coefficient de Bhattacharyya.
3. Finalement, les Gaussiennes similaires sont regroupées grâce à l'algorithme du Clustering Spectral (CS) et les groupes de données finaux sont obtenus.

La suite de cette section est organisée comme suit. Le concept du mélange Gaussien est expliqué et détaillé en sous-section 5.4.2. La sous-section 5.4.3 montre comment utiliser le coefficient de Bhattacharyya afin de combiner certaines distributions normales similaires, grâce au CS. Des résultats expérimentaux préliminaires sont donnés en sous-section 5.4.5. Comme dit plus haut, la méthode GSC étant encore dans sa forme préliminaire, seules des données synthétiques sont utilisées. Enfin, la sous-section 5.4.6 présente deux voies d'amélioration possibles pour l'algorithme proposé.

5.4.2 Mélange Gaussien

Généralités

Un modèle de mélange est un modèle probabiliste qui approxime la densité d'un jeu de données par une somme pondérée de distributions de probabilité du même type mais paramétrées différemment. Nous nous concentrons ici sur les MMG, c'est-à-dire que les distributions précédentes sont supposées normales.

Plus généralement, un MMG $\mathcal{M}$ à K composantes peut être défini comme suit :

$$\mathcal{M} = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\boldsymbol{\mu_k}; \boldsymbol{\Sigma_k}), \tag{5.48}$$

où π_k est le poids associé à la $k^{\text{ème}}$ composante avec $\sum_{k=1}^{K} \pi_k = 1$ et $\mathcal{N}(\boldsymbol{\mu_k}; \boldsymbol{\Sigma_k})$ la loi normale en dimension d de moyenne $\boldsymbol{\mu_k}$ et de matrice de covariance $\boldsymbol{\Sigma_k}$. On peut également exprimer la probabilité qu'une donnée $\boldsymbol{x} \in \mathbb{R}^d$ soit générée par la composante k par :

$$p(\boldsymbol{x}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma_k}|^{1/2}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu_k})^T \boldsymbol{\Sigma_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_k})}{2}\right). \tag{5.49}$$

Une application imaginable en une dimension est la modélisation de l'histogramme d'une image en niveaux de gris dans l'optique d'une segmentation [Lai+12]. A titre d'illustration, considérons la fameuse image Lena représentée en Figure 5.9a. L'histogramme de cette image convertie en niveaux de gris est affiché en bleu en Figure 5.9c. On y donne également en rouge le MMG à sept composantes associé. Suite à cette modélisation, on peut distinguer sept catégories (via les lignes verticales noires sur le graphe) et remplaçer chaque niveau de gris par la moyenne de la composante à laquelle il est associé. L'image segmentée est donnée en Figure 5.9b.

Algorithme Espérance-Maximisation

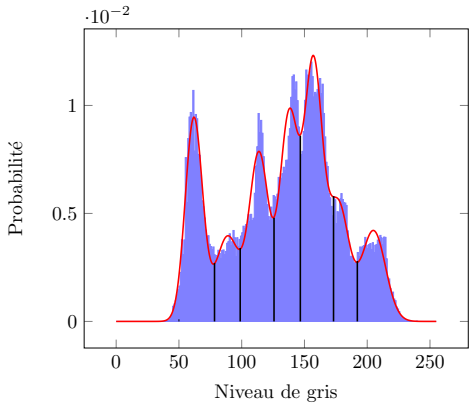
Si le nombre de composantes K est connu, il est possible de déterminer un MMG, tel que celui de l'Équation 5.48, uniquement à partir des données $\{\boldsymbol{x_i} \in \mathbb{R}^d, i = 1, \ldots, n\}$. Pour ce faire, on utilise l'algorithme Espérance-Maximisation (EM), *Expectation-Maximization* en anglais, développé par Dempster *et al.* en 1977 [Dem+77]. Ce dernier consiste à déterminer, de manière itérative, les paramètres du mélange Gaussien de telle sorte que la fonction de vraisemblance associée au modèle soit maximisée. Les paramètres que nous mentionnons sont le poids π_k, la moyenne $\boldsymbol{\mu_k}$ et la matrice de covariance $\boldsymbol{\Sigma_k}$ de chaque composante k.

Concrètement, l'algorithme EM se décompose en quatre étapes :

1. **Initialisation des paramètres**
   Les approches classiques fixent $\pi_1 = \cdots = \pi_K = 1/K$ et $\boldsymbol{\Sigma_1} = \cdots = \boldsymbol{\Sigma_K} = \boldsymbol{\Sigma}$, où $\boldsymbol{\Sigma}$ est la matrice de covariance du jeu de données complet. Quant aux moyennes, elles sont couramment initialisées par des points choisis aléatoirement.

(a) Image originale.

(b) Image segmentée.



(c) Histogramme des niveaux de gris (en bleu) et modélisation par mélange Gaussien (en rouge).

Figure 5.9. Segmentation de l'image Léna grâce à un MMG.

2. **Etape E(spérance)**
   On calcule ici la probabilité γ_{ik} que la donnée i soit générée par la composante k. Ainsi :

$$\forall (i,k) \in [\![1;n]\!] \times [\![1;K]\!] \qquad \gamma_{ik} = \frac{\pi_k \, \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_k};\boldsymbol{\Sigma_k})}{\sum_{j=1}^{K} \pi_j \, \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_j};\boldsymbol{\Sigma_j})} \tag{5.50}$$

3. **Etape M(aximisation)**
   Les paramètres sont mis à jour grâce aux probabilités déterminées à l'étape précédente. Pour chaque composante k :

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ik} \tag{5.51}$$

$$\boldsymbol{\mu_k} = \frac{\sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x_i}}{\sum_{i=1}^{n} \gamma_{ik}} \tag{5.52}$$

$$\boldsymbol{\Sigma_k} = \frac{\sum_{i=1}^{n} \gamma_{ik}(\boldsymbol{x_i} - \boldsymbol{\mu_k})^T(\boldsymbol{x_i} - \boldsymbol{\mu_k})}{\sum_{i=1}^{n} \gamma_{ik}} \tag{5.53}$$

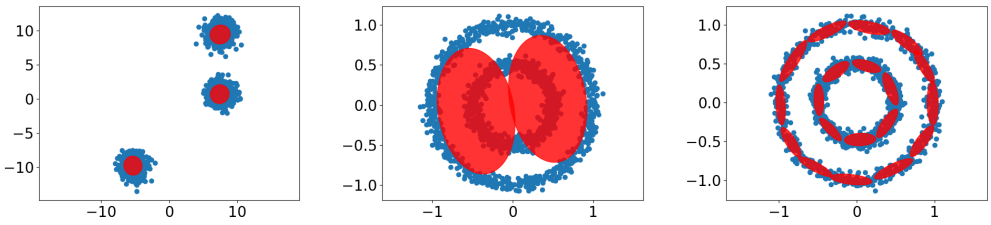
4. **Répétition jusqu'à convergence**

Grand nombre de variantes de cette méthode ont été développées au fil des années : c'est le cas des algorithmes Stochastic Expectation Maximization (SEM) [CD85], Classification Expectation Maximization (CEM) [CG92] ou encore α-EM [Mat03].

Toutefois, des difficultés apparaissent avec cette technique du fait de la malédiction de la dimension (voir section 4.1). En particulier, la distance de Mahalanobis présente à l'intérieur de l'exponentielle dans l'Équation 5.49 souffre de ce phénomène du fait de la matrice de covariance, dont le nombre de termes évolue proportionnellement au carré de la dimensionalité. Ruan *et al.* [Rua+11] ont proposé une modification de l'algorithme EM afin de contrecarrer ce problème. L'idée est d'ajouter une régularisation de type ℓ_1 au niveau de l'étape M. Plus précisément, après le calcul de la matrice de covariance de chaque composante, la méthode *graphical Lasso* [Fri+08] est appliquée à chacune de ces matrices. De cette manière, les matrices de covariance sont rendues parcimonieuses, c'est-à-dire incluant beaucoup d'éléments égaux à zéro, et les effets de la malédiction de la dimension sont estompés.

Partitionnement de données par mélange Gaussien

En plus de sa capacité à approcher la distribution d'un jeu de données, le partitionnement par MMG est également possible. Par exemple, les trois *blobs* représentés par des points bleus en Figure 5.10a peuvent facilement être partitionnés grâce à un modèle à trois composantes : les ellipses de confiance en rouge correspondent aux matrices de covariance des différentes composantes. Chaque donnée est donc associée à une et une seule Gaussienne et les trois clusters sont récupérés.



(a) 3 composantes. **(b)** 2 composantes. **(c)** 19 composantes.

Figure 5.10. Jeux de données bidimensionnelles et MMG associés.

Toutefois, cette approche de clustering échoue dans les cas où les différents groupes de données ne sont pas “sphériques”. En effet, quand appliquée à deux cercles concentriques de rayons différents (voir Figure 5.10b), la méthode manque complètement la distribution des données et influence le partitionnement qui suit dans la mauvaise direction.

Plusieurs remarques sont à faire ici. D’abord, ce problème apparait dans le cas où les véritables clusters (en l’occurence les deux anneaux) partagent la même moyenne. De plus, le nombre de composantes est choisi comme étant égal au nombre de clusters, *i.e.* deux. Cela implique que le nombre de groupes de données est disponible avant exécution de l’algorithme, ce qui semble utopique en pratique. Cela montre également que choisir le nombre de composantes égal au nombre de clusters est souvent inapproprié.

Modèle de mélange optimal

Comme évoqué précédemment, le choix du nombre de composantes d’un MMG est critique et souvent indépendant du nombre de groupes présents dans les données. Nous proposons ici d’effectuer une recherche exhaustive du modèle optimal. Plus précisément, différents modèles (ou, de manière équivalente, différents nombres de composantes) sont essayés sur les données en entrée et celui qui les représente au mieux est considéré comme idéal.

Une question qui vient naturellement est le choix du critère qui permet une sélection de modèle. De nombreuses techniques ont été développées au fil des années pour traiter ce sujet [MR14]. Nous proposons ici d’utiliser le Bayesian Information Criterion (BIC) [Sch78] car **(a)** il ne sous-estime pas asymptotiquement (*i.e.* quand la taille du jeu de données considéré augmente) le nombre de véritables composantes [Ler92] et **(b)** l’estimation de densité résultante est cohérente avec la vérité-terrain [RW97 ; Ker00]. Ce dernier est défini comme suit :

$$BIC = t \ln n - 2 \ln \mathcal{L}, \tag{5.54}$$

où n est le nombre de points du jeu de données considéré et $\mathcal{L}$ est la fonction de vraisemblance associée au modèle. Quant à t, c’est le nombre de paramètres à estimer : par exemple, dans le cas d’un MMG à quatre composantes en trois dimensions, t est égal à $4 \times 10 = 40$ car il faut déterminer, pour chaque composante, un paramètre pour le poids, trois pour la moyenne et six pour la matrice de covariance.

Le modèle qui engendre la valeur BIC la plus faible est considéré comme optimal, car il représente le mieux les données originales. En particulier, dans le cas des deux cercles concentriques mentionnés ci-dessus, on peut visualiser l'évolution du critère en fonction du nombre de composantes. On observe alors, sur la Figure 5.11, que le minimum est atteint pour 19 composantes. En effet, il est clair que, dans ce cas, le modèle obtenu est une meilleure approximation de la distribution des données que pour un modèle à deux composantes (voir Figure 5.10 au milieu et à droite). Au regard de l'Équation 5.54, on comprend que ce critère est un compromis entre la

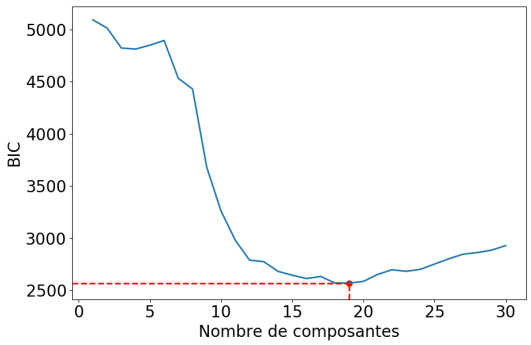


Figure 5.11. Evolution du BIC en fonction du nombre de composantes du MMG dans le cas de deux cercles concentriques. Le minimum est représenté en rouge.

qualité d'approximation des données, représentée par la fonction de vraisemblance $\mathcal{L}$, et la complexité du modèle, matérialisée par son nombre t de paramètres.

5.4.3 Clustering spectral

La principale conséquence de la détermination du MMG optimal à travers la minimisation du BIC, comme vu précédemment, réside dans le fait que le nombre de composantes obtenu peut ne pas être indicatif du véritable nombre de clusters. En effet, étant donné qu'une distribution normale ne peut représenter précisément qu'un groupe de données elliptique, un cluster dont la forme est plus complexe est approchée par plusieurs Gaussiennes. Nous sommes alors dans un cas de **surapprentissage** : le nombre de composantes du MMG résultant est supérieur ou égal au nombre de clusters présents dans les données.

Revenons un instant sur l'exemple des deux cercles concentriques de la Figure 5.10. Le modèle de mélange à 19 composantes semble intéressant, bien que ce jeu de données ne comporte que deux clusters. Toutefois, notons que les Gaussiennes du

cercle extérieur présentent des similarités paire à paire, de même pour le cercle intérieur, alors qu'elles sont relativement différentes entre les deux cercles.

Ce qui suit vise donc à fusionner intelligemment les Gaussiennes afin d'obtenir les vrais clusters. Pour ce faire, nous commençons par définir la notion d'affinité entre lois normales via le coefficient de Bhattacharyya, puis ces dernières sont fusionnées de manière automatique grâce à l'algorithme du Clustering Spectral (CS).

Similarité entre Gaussiennes

Afin de mesurer la ressemblance entre distributions Gaussiennes, nous introduisons ici les distance et coefficient de Bhattacharyya [Bha43]. Considérons deux distributions continues p et q. On peut alors déterminer la distance de Bhattacharyya d_B entre p et q par :

$$d_B(p,q) = -\ln\left(\int \sqrt{p(x)q(x)}\mathrm{d}x\right) \tag{5.55}$$

L'intégrale à l'intérieur du logarithme népérien est appelée **coefficient de Bhattacharyya**. Une simplification existe dans le cas de distributions normales. Supposons $p \sim \mathcal{N}(\boldsymbol{\mu_p}; \boldsymbol{\Sigma_p})$ et $q \sim \mathcal{N}(\boldsymbol{\mu_q}; \boldsymbol{\Sigma_q})$. Alors :

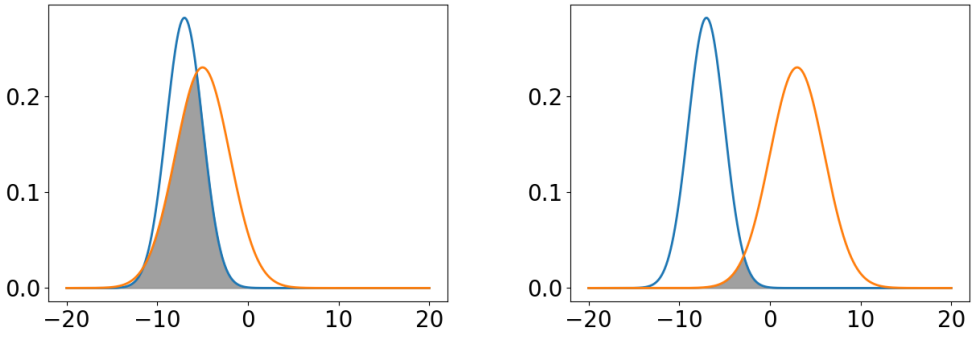
$$d_B(p,q) = \frac{1}{8}(\boldsymbol{\mu_p} - \boldsymbol{\mu_q})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_p} - \boldsymbol{\mu_q}) + \frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}|}{\sqrt{|\boldsymbol{\Sigma_p}||\boldsymbol{\Sigma_q}|}}, \tag{5.56}$$

où $|.|$ représente le déterminant et $\boldsymbol{\Sigma}$ est la moyenne arithmétique des deux matrices de covariance, en d'autres mots $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma_p} + \boldsymbol{\Sigma_q})/2$. On peut aisément en déduire le coefficient de Bhattacharyya c_B par :

$$c_B(p,q) = \exp\left(-d_B(p,q)\right) \tag{5.57}$$

Ce coefficient possède une interprétation géométrique : en effet, ce dernier constitue une approximation du taux de recouvrement de deux distributions statistiques. Il approche 1 quand les deux distributions comparées sont quasi-identiques, 0 dans le cas contraire. Cette interprétation est illustrée par deux exemples simples en Figure 5.12.

Considérons maintenant le cas où, d'après un jeu de données spécifique, un MMG optimal à C composantes est obtenu, comme expliqué précédemment. Nommons



(a) c_B = 0,8897. **(b)** c_B = 0,1404.

Figure 5.12. Interprétation géométrique du coefficient de Bhattacharyya comme approximation du taux de recouvrement entre deux Gaussiennes.

ces différentes composantes $\mathcal{N}_1, \ldots, \mathcal{N}_C$. On peut alors construire la matrice de similarité $\boldsymbol{S} = (S_{ij})$, de taille $C \times C$, telle que

$$S_{ij} = \begin{cases} c_B\left(\mathcal{N}_i, \mathcal{N}_j\right) & \text{si } i \neq j \\ 0 & \text{si } i = j \end{cases} \quad \forall (i,j) \in [\![1; C]\!]^2. \tag{5.58}$$

Cette matrice représente donc les coefficients de Bhattacharyya paire à paire, autrement dit la similarité entre chaque paire de Gaussiennes. Elle est symmétrique et est nulle le long de la diagonale.

Fusion par clustering spectral

Dans l'optique d'un clustering, l'idée qui vient immédiatement consiste à partitionner la matrice de similarité, définie en Équation 5.58, de telle sorte que des ensembles de distributions normales, dont le taux de recouvrement mutuel est significatif, soient déterminés. En d'autres termes, nous souhaitons découvrir des clusters au sein desquels il existe un chemin évident d'une Gaussienne à l'autre, soit directement, soit à travers d'autres Gaussiennes du même groupement.

Afin de procéder à cette fusion, nous optons pour le Clustering Spectral (CS) [Lux07], et plus particulièrement l'algorithme proposé par Andrew Ng en 2001 [Ng+01]. En introduisant n_c le nombre de clusters à découvrir, on peut appliquer ce dernier comme suit :

1. Normaliser la matrice de similarité $\boldsymbol{S}$ par $\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{S}\boldsymbol{D}^{-1/2}$. $\boldsymbol{D}$ est la matrice diagonale somme ligne par ligne de $\boldsymbol{S}$, *i.e.* $\boldsymbol{D} = (D_{ij})$ avec

$$D_{ij} = \begin{cases} \sum_{j=1}^{C} S_{ij} & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \quad \forall (i,j) \in [\![1; C]\!]^2. \tag{5.59}$$

   Rappelons que C est le nombre de composantes du MMG optimal déterminé précédemment.

2. Décomposer $\boldsymbol{L}$ en éléments propres et récupérer ses vecteurs propres $\boldsymbol{V} \in \mathbb{R}^{C \times C}$. On assume dans la suite qu'ils sont triés dans l'ordre décroissant des valeurs propres.
3. Construire $\tilde{\boldsymbol{V}}$ en ne gardant que les n_c premiers vecteurs propres. $\tilde{\boldsymbol{V}}$ est donc de taille $C \times n_c$.
4. Obtenir $\boldsymbol{Y}$ en normalisant les lignes de $\tilde{\boldsymbol{V}}$.
5. Appliquer l'algorithme K-means [AV07] en considérant chaque ligne de $\boldsymbol{Y}$ comme une donnée.

Apportons ici quelques précisions. La méthode précédente est appliquée à la matrice de similarité $\boldsymbol{S}$ de l'Équation 5.58, qui regroupe les coefficents de Bhattacharyya paire à paire entre les distributions normales obtenues par le MMG optimal. Ainsi, l'algorithme du CS opère sur les composantes du modèle et non directement sur les différents points du jeu de données, comme c'est le cas habituellement. Par suite, si le point $\boldsymbol{x_i}$ original appartient à la composante n°3 et si cette dernière est associée au cluster n°5, alors $\boldsymbol{x_i}$ correspond au cluster n°5. Il en va de même pour toutes les données de chaque distribution originale, ainsi que pour l'ensemble des individus du jeu de données.

La difficulté majeure de l'algorithme du CS comme expliqué ci-dessus est qu'il requiert le nombre n_c de clusters avant exécution. Cela implique une interaction avec l'utilisateur et la connaissance du nombre de groupes cachés dans les données, ce qui est impensable dans de nombreuses situations réelles, surtout en grande dimension.

Afin de surmonter cette problématique, différentes idées ont été proposées, *e.g.* [ZMP04]. Dans le cas de l'algorithme GSC détaillé ici, nous proposons d'itérer les étapes 3 – 4 – 5 de l'algorithme CS : l'idée consiste à fournir une recherche exhaustive du nombre de clusters, de 1 à C. A l'instar de ce qui a été fait en sous-section 5.4.2, l'optimalité est jugée par la sortie de l'algorithme K-means sous la forme de la somme des distances Euclidiennes quadratiques de chaque donnée au barycentre du cluster auquel elle appartient, également appelée **distorsion**. Plus précisément, si

l'on considère un jeu de données $\{\boldsymbol{x_i}\}$ partitionné en n_c clusters et $\boldsymbol{\mu_j}$ les différents barycentres, alors la distorsion est égale à

$$\sum_{j=1}^{n_c} \sum_{\boldsymbol{x_i} \in \text{cluster } j} \|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|_2^2. \tag{5.60}$$

Le nombre optimal de clusters est celui qui, entre 1 et C, donne la plus faible distorsion.

A ce niveau, nous souhaitons souligner deux éléments importants de cette étape de recherche exhaustive. D'abord, cette idée semble présenter certaines ressemblances avec les mesures internes de validation de partitionnement [Liu+13b]. Rappelons que ces dernières visent à comparer deux partitionnements, par exemple obtenus lors de deux exécutions d'un même algorithme sur un même jeu de données, sans aucune vérité-terrain. Toutefois, la situation est quelque peu différente dans notre cas, étant donné que les données évoluent à chaque itération, notamment la dimension. C'est une dissimilitude notoire avec l'algorithme X-means proposé par Pelleg et Moore [PM00] où les données restent les mêmes lors des nombreuses exécutions. De plus, comme dit précédemment, il est presque impossible en pratique de deviner le nombre exact de clusters. Cependant, dans le cas de l'approche proposée dans cette section, on peut inclure des informations *a priori* sous la forme de plage de valeurs, aussi bien pour le nombre de composantes que pour celui de clusters, élément qui semble plus adapté expérimentalement, en plus de permettre une accélération logicielle.

5.4.4 Résumé

Nous décrivons dans cette sous-section un résumé de la méthode proposée. En supposant un jeu de données en entrée, l'algorithme GSC fonctionne alors en trois temps :

1. D'abord, le MMG qui approche au mieux la distribution des données est déterminé. Pour ce faire, différents nombres de composantes sont testés et celui qui aboutit au BIC le plus faible est considéré comme optimal.

2. Ensuite, la matrice de similarité $\boldsymbol{S}$, contenant les coefficients de Bhattacharyya entre chaque paire de Gaussiennes du modèle de mélange précédent, est construite.

3. Enfin, on applique l'algorithme du CS sur cette matrice $\boldsymbol{S}$ afin de fusionner de manière efficace les distributions normales présentant des similarités. De

même, le nombre de clusters optimal est déterminé via minimisation de la distorsion suite à l'exécution de K-means.

Au final, la technique GSC est totalement non-paramétrique. On peut toutefois ajouter des contraintes au niveau des deux recherches exhaustives (pour les nombres de composantes et de clusters optimaux). Il est également possible de tirer une variante de cet algorithme où le nombre de groupes à découvrir est fourni en entrée.

5.4.5 Résultats expérimentaux

Cette sous-section est consacrée aux expériences menées. Elle présente une comparaison avec d'autres algorithmes de partitionnement de données, notamment celui traditionnel par mélange Gaussien. Nous comparons également les deux variantes de GSC, à savoir la version non-paramétrique et celle dans laquelle le nombre de clusters à découvrir est donné comme paramètre d'entrée.

Jeux de données et métrique

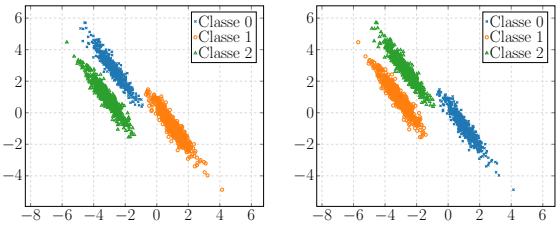
De même qu'en sous-section 5.3.8, les données synthétiques du CLUSTERING BENCHMARK [3] sont utilisées. Les partitionnements prédits sont comparés à la vérité-terrain via le score de Fowlkes-Mallows [FM83]. De plus, pour chaque jeu de données et chaque méthode, une recherche exhaustive des paramètres optimaux est effectuée. Enfin, en ce qui concerne notre méthode, la plage pour le nombre de composantes du MMG est fixée à $[1; 75]$.

Comparaison avec le clustering par mélange Gaussien traditionnel

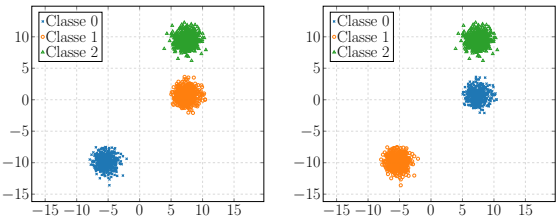
En tant que préliminaire, la Figure 5.13 présente une comparaison visuelle entre l'algorithme proposé et l'approche par mélange Gaussien classique sur quelques jeux de données synthétiques. On peut y observer des résultats identiques sur les deux premiers exemples (Figure 5.13a et Figure 5.13b). En revanche, notre méthode (à gauche) domine dans les deux autres cas (Figure B.1a et Figure 5.13d). L'algorithme de partitionnement de données par MG est incapable de découvrir des clusters de formes plus complexes, c'est-à-dire non-elliptiques. Nous souhaitons également souligner que, contrairement au clustering par MG pour lequel le nombre de clusters

3. https://github.com/deric/clustering-benchmark

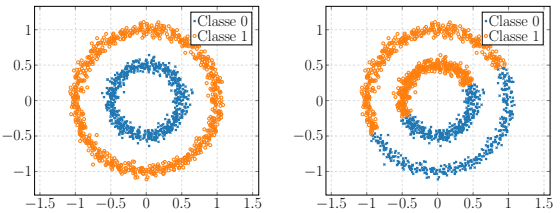
doit être spécifié par l'utilisateur, GSC découvre précisément le nombre correct, et ce de manière automatique.



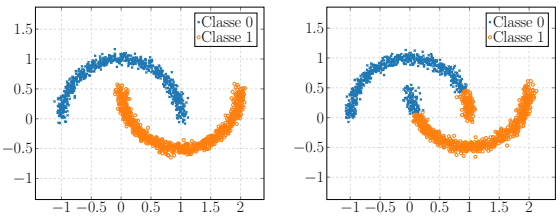
(a) Blobs anisotropiques.



(b) Blobs.



(c) Cercles concentriques.



(d) Demi-lunes.

Figure 5.13. Comparaison entre GSC (colonne de gauche) et clustering par mélange Gaussien classique (colonne de droite) sur quatre jeux de données synthétiques en deux dimensions. Chaque marqueur/couleur représente une classe différente.

Comparaison des deux variantes de GSC

Nous présentons ici les résultats obtenus par l'approche proposée dans deux situations : **(a)** quand le nombre de clusters est donné en entrée, version que nous

nommons GSC_vt dans la suite (vt pour vérité-terrain), et **(b)** quand la méthode est totalement non-paramétrique, nommée GSC_np. Pour ce faire, nous utilisons l'histogramme des scores de Fowlkes-Mallows calculés sur l'ensemble des bases de données. La Figure 5.14 affiche la superposition des deux histogrammes ainsi que des moyennes (en pointillés).

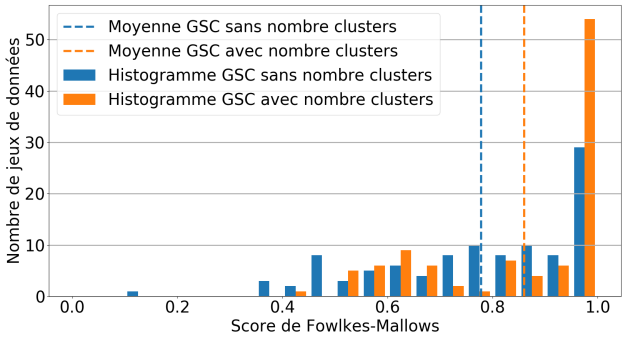


Figure 5.14. Histogrammes des scores de Fowlkes-Mallows pour les deux variantes de l'algorithme GSC, à savoir avec et sans le nombre de clusters donné en entrée. Les moyennes respectives sont affichées en pointillés.

Comme attendu, la variante GSC_vt surclasse GSC_np. En effet, la moyenne sur l'ensemble des jeux de données est égale à 0,860 pour GSC_vt contre 0,778 dans le cas de la version non-paramétrique. Cette différence provient notamment des itérations de l'algorithme K-means (*cf.* sous-section 5.4.4) et peut s'expliquer par trois facteurs :

1. K-means est appliqué sur un faible nombre de points, compris entre 1 et C, où C correspond au nombre de composantes du MMG optimal. Il est donc complexe d'obtenir un partitionnement cohérent.
2. La distorsion ne prend en compte que la distance entre chaque donnée et le barycente du groupe auquel elle appartient. Le nombre de clusters découverts ainsi que la dimensionalité, paramètres qui évoluent durant ces itérations, ne rentrent pas en jeu.
3. Le cas limite où l'algorithme K-means est exécuté avec le même nombre de clusters que le nombre de points aboutit à une distorsion nulle : cette solution est donc considérée comme optimale, peu importe le jeu de données, bien qu'incorrecte.

Comparaison avec d'autres algorithmes de clustering

Dans le but de fournir une comparaison équitable, nous nous mettons dans les conditions où le nombre de clusters est connu en avance, étant donné que c'est

un argument obligatoire pour les algorithmes de partitionnement de données par MG, K-means [AV07] et CS [Ng+01]. Nous procédons comme suit : la plupart des méthodes précédentes n'étant pas déterministes, elles sont exécutées sur chaque base de données, répétées 50 fois pour assurer la stabilité des résultats et le score de Fowlkes-Mallows moyen est gardé. Le Tableau 5.2 présente un extrait des résultats obtenus sur 23 jeux de données uniquement. Le meilleur score de chaque ligne est affiché en gras et la moyenne de chaque méthode est donnée sur la dernière ligne.

Jeu de données	GSC	DBSCAN	MG	K-means	CS
2D-20C-NO0	0,987	0,971	0,948	0,966	**0,992**
2D-3C-NO123	**0,981**	0,890	0,937	0,821	0,941
2D-4C-NO4	**1**	0,963	0,899	0,982	0,992
2D-4C-NO9	**1**	0,896	0,992	0,924	0,975
2D-4C	**1**	**1**	**1**	**1**	**1**
CURVES1	**1**	**1**	0,499	0,499	**1**
CURVES2	0,907	0,376	0,905	0,947	**0,952**
DARTBOARD2	0,570	**0,925**	0,547	0,547	0,548
DONUT1	**0,996**	0,991	0,526	0,505	0,992
ELLY-2D10C13S	**0,937**	0,707	**0,937**	0,908	0,929
ENGYTIME	0,614	0,732	0,690	0,736	**0,970**
PATHBASED	0,692	**0,980**	0,856	0,798	0,979
PMF	0,992	0,925	0,976	**0,993**	**0,993**
SPHERICAL_5_2	**1**	**1**	0,989	**1**	**1**
SPHERICAL_6_2	**1**	**1**	0,528	0,516	**1**
SQUARE1	0,941	0,499	0,941	0,937	**0,947**
SQUARE2	0,893	0,499	0,892	0,894	**0,898**
TETRA	0,650	**0,707**	0,596	0,574	0,705
TWENTY	**1**	0,707	**1**	**1**	**1**
TWODIAMONDS	0,865	0,947	0,932	0,930	**1**
WINGNUT	0,995	0,942	0,995	0,995	**0,999**
ZELNIK3	**1**	**1**	**1**	0,647	**1**
ZELNIK5	0,810	**1**	0,789	0,791	0,808
Moyenne	0,860	0,852	0,789	0,760	**0,884**

Table 5.2. Extrait (23 bases de données) des scores de Fowlkes-Mallows obtenus par notre méthode ainsi que quatre autres. La moyenne sur l'ensemble des données est reportée en dernière ligne.

La méthode proposée montre de meilleures performances que le partitionnement par MG et K-means. En effet, dans le cas où le nombre de clusters est connu et fourni en entrée des deux algorithmes, GSC est capable de déterminer les différents groupes 9,1% mieux que l'approche par MG. Les deux techniques donnent des résultats similaires dans le cas de clusters elliptiques. Il est également intéressant de souligner que le clustering par MG offre de meilleures performances que K-means, étant donné qu'il généralise K-means à des clusters ellipsoïdaux.

Dans l'optique de démontrer les meilleures performances de GSC sur le clustering par MG, nous réalisons un test statistique sur la différence des deux moyennes. Sur les $n = 101$ bases de données testées, on détermine les moyennes ($\mu_{GSC} = 0,860$, $\mu_{MG} = 0,789$) et les écarts-types ($\sigma_{GSC} = 0,171$, $\sigma_{MG} = 0,197$) des deux méthodes. On peut alors considérer les hypothèses nulle $\mathcal{H}_0$ et alternative $\mathcal{H}_1$ suivantes :

$$\mathcal{H}_0 : \mu_{MG} = \mu_{GSC} \tag{5.61}$$

$$\mathcal{H}_1 : \mu_{MG} < \mu_{GSC} \tag{5.62}$$

Par suite, on peut calculer la statistique z comme suit :

$$z = \frac{\mu_{MG} - \mu_{GSC}}{\sqrt{\sigma_{MG}^2/n + \sigma_{GSC}^2/n}} = -2,735. \tag{5.63}$$

Si l'on considère u une variable aléatoire suivant la loi normale centrée-réduite $\mathcal{N}(0;1)$, alors $\mathbb{P}(u < 2,735) = 0,9968 = 1 - 0,0032$. Ceci nous permet donc de conclure au rejet de l'hypothèse $\mathcal{H}_0$ avec une valeur p égale à 0,32%.

GSC montre de meilleures performances que DBSCAN [Est+97 ; Sch+17] mais est dépassé par CS. Il est toutefois important de garder à l'esprit que plusieurs paramètres sont à fixer pour ces méthodes :

- Pour DBSCAN, le rayon ε des voisinages sphériques ainsi que le nombre $MinPts$ de données qu'un voisinage doit contenir afin d'être considéré comme valide.
- Pour CS, l'écart-type du noyau Gaussien ainsi que le nombre de clusters à découvrir.

De plus, une recherche exhaustive sur les paramètres est conduite et seul le meilleur score de Fowlkes-Mallows est affiché. Un tel processus est impraticable en conditions réelles. Rappelons que, pour cette comparaison, seul le nombre de clusters est fourni et notre méthode est automatique hormis cet unique paramètre.

En résumé, notre modèle donne des résultats compétitifs dans la plupart des cas sauf trois, à savoir DARTBOARD2, ENGYTIME and TETRA. Ces mauvais scores sont principalement causés par des MMG imprécis dus à des clusters qui se chevauchent fortement (voir Figure 5.15) ou des groupes comportant trop peu de points.

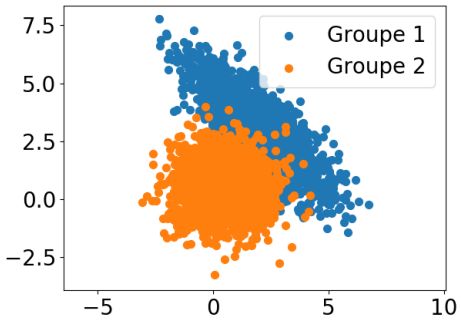


Figure 5.15. Jeu de données ENGYTIME issu du CLUSTERING BENCHMARK. On peut noter qu'un chevauchement existe entre les deux clusters.

5.4.6 Améliorations possibles du modèle

Nous présentons ici deux pistes d'amélioration de la méthode proposée dans cette section : la première se focalise sur l'étape d'obtention du modèle de mélange idéal, la deuxième sur une approche probabiliste de GSC. Ces deux idées ont pu être mises en pratique et, même si elles n'apportent pas les résultats espérés, nous estimons qu'il est tout de même intéressant de les mentionner et de les détailler.

Modèle de mélange optimal : approche par problème du bandit

Cette première idée concerne la détermination du MMG optimal, comme expliquée en sous-section 5.4.2. Pour rappel, l'objectif était alors de calculer **un seul** modèle pour chaque nombre de composantes dans une plage prédéfinie, puis de choisir celui qui aboutissait au critère (ici BIC) le plus faible.

Cependant, on peut remarquer que la valeur du BIC fluctue si l'opération de modélisation est répétée pour un **même** nombre de composantes. Cela peut s'expliquer notamment par le fait que l'initialisation du modèle de mélange est aléatoire. Deux remarques en découlent :

— le modèle optimal n'est pas unique, ce qui implique des valeurs du BIC proches, mais pas égales.

— il existe des risques de tomber dans des minima locaux et d'obtenir ainsi un modèle aberrant, non représentatif des données.

A titre d'illustration, considérons à nouveau l'exemple des deux cercles concentriques de la Figure 5.10. L'histogramme des valeurs du critère BIC, obtenues en répétant l'opération de modélisation par MG à 19 composantes un grand nombre de fois (ici 5000), est représenté en Figure 5.16. On peut y observer que les valeurs obtenues sont diverses. Ainsi, la courbe que l'on peut obtenir en déterminant le BIC pour différents nombres de composantes (voir Figure 5.11) n'est pas déterministe : le minimum peut donc varier, il en est de même pour les traitements ultérieurs.

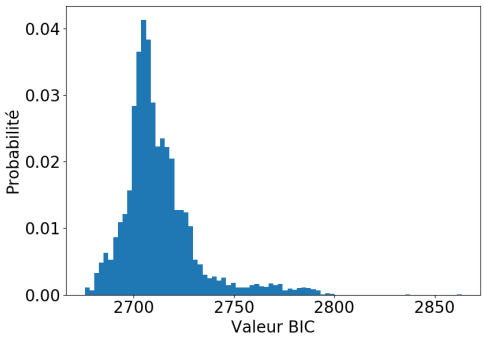


Figure 5.16. Histogramme des valeurs BIC obtenues suite à 5000 répétitions de la modélisation par MG à 19 composantes sur les deux cercles concentriques.

Plusieurs solutions sont imaginables pour contrer cette difficulté. La première consiste à travailler sur la robustesse de l'initialisation du MMG (voir [BB16] par exemple) : bien que réduit, l'aspect aléatoire y est toutefois toujours présent. Une autre solution vise à répéter l'algorithme EM un grand nombre de fois pour chaque nombre de composante, afin de stabiliser le BIC : cette approche semble cependant coûteuse en termes de calculs.

La démarche que nous proposons ici repose sur le problème du bandit manchot. Ce dernier peut être expliqué de manière vulgarisée comme suit. Considérons un utilisateur devant plusieurs machines à sous, chacune délivrant une récompense régie par une loi de probabilité, évidemment inconnue du joueur. L'objectif est pour l'utilisateur de deviner quelle(s) machine(s) jouer dans l'optique de maximiser son gain.

L'analogie avec notre situation est la suivante : on peut considérer les différents nombres de composantes du MMG comme étant les machines. Chaque essai nous donne une récompense, dans notre cas une valeur du critère BIC (en réalité l'opposé de cette dernière car nous cherchons ici à minimiser la récompense, pas à la maximiser). Par la suite, il est possible de déterminer le nombre de composantes

qui engendre une valeur de BIC minimale en moyenne : on trouve alors le MMG optimal, sans avoir à essayer tous les nombres de composantes différents plusieurs fois.

La version utilisée en pratique pour la résolution du problème du bandit est l'algorithme Upper Confidence Bounds (UCB) [Aue+02]. De plus, nous considérons dans la suite uniquement le cas où le nombre de clusters est connu.

Suite aux expérimentations, il s'avère que cette approche ne permet pas d'améliorer les résultats, au contraire. La moyenne sur l'ensemble des bases de données tombe à 0,821, ce qui équivaut à une baisse de 4%. On peut expliquer principalement cette chute par le fait qu'un critère d'arrêt sur la récompense est fixé : en général, l'algorithme UCB est exécuté pendant un grand nombre d'itérations, précisé avant exécution. Ce nombre étant difficile à choisir et dû à des raisons pratiques de temps d'exécution, un critère d'arrêt semble plus approprié. Toutefois, le MMG obtenu n'est pas optimal et induit des erreurs dans les processus suivants.

Quoi qu'il en soit, bien que les résultats ne soient pas concluants, nous pensons que l'idée d'une optimisation afin de déterminer le MG idéal est intéressante. Il parait légitime de penser que des améliorations peuvent encore être apportées. On peut notamment penser à une autre variante pour la résolution du problème du bandit, plus adaptée à la situation. Une deuxième solution consisterait à procéder à une descente de gradient afin de minimiser le BIC et déterminer le nombre de composantes optimal automatiquement, sans recherche exhaustive.

Approche probabiliste

On peut résumer l'algorithme GSC comme suit de manière grossière. La première étape donne le degré d'appartenance, sous la forme d'une probabilité, de chaque donnée à chaque distribution normale du modèle. Ces Gaussiennes sont ensuite partitionnées via une méthode *hard* (typiquement K-means) : ce terme indique que le résultat du clustering est un ensemble d'étiquettes **entières**, spécifiant l'appartenance d'une composante du MMG à tel ou tel groupe.

Le problème est le suivant : imaginons un point $\boldsymbol{x}$ d'un jeu de données spécifique, approché par un modèle de mélange à trois composantes $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$. Supposons de plus que $\boldsymbol{x}$ appartienne à $\mathcal{N}_1$ (respectivement $\mathcal{N}_2$ et $\mathcal{N}_3$) avec une probabilité égale à 0,36 (respectivement 0,32 et 0,32). Si la composante $\mathcal{N}_1$ est associée au cluster n°1, alors la donnée $\boldsymbol{x}$ l'est également. On se rend bien compte sur cet exemple que, les probabilités d'appartenance à chaque composante étant trop proches, le clustering

hard, via un seuillage par le maximum par exemple, peut engendrer des erreurs irréversibles. Pour simplifier, c'est exactement comme si les informations apportées par l'algorithme EM étaient ignorées.

Ainsi, nous proposons de remplacer la dernière étape de GSC, c'est-à-dire l'application de K-means sur les différentes composantes du modèle, par une méthode de clustering *soft* qui, contrairement à la précédente, retourne, non pas une étiquette entière, mais une probabilité d'appartenance à tel ou tel cluster. Nous utilisons ainsi la version floue de K-means : Fuzzy C-Means (FCM) [Dun73 ; Bez81].

L'idée générale est donc de combiner les différentes probabilités, celles liées au mélange Gaussien et celles du partitionnement flou, pour plus de robustesse. Comment procéder ? Considérons la situation simpliste d'une donnée $\boldsymbol{x}$, d'un MMG à trois composantes et d'un partitionnement final en deux groupes. Cette dernière est représentée en Figure 5.17 sous la forme d'un arbre de probabilité illustrant les différents cas possibles. Les probabilités situées sur la gauche de l'arbre sont issues de la modélisation par MG, celles à droite proviennent de l'algorithme FCM. On peut alors calculer la probabilité que $\boldsymbol{x}$ appartienne au cluster n°1 grâce à la formule des probabilités totales :

$$p(C_1|\boldsymbol{x}) = p(\mathcal{N}_1|\boldsymbol{x}) \times p(C_1|\mathcal{N}_1) + p(\mathcal{N}_2|\boldsymbol{x}) \times p(C_1|\mathcal{N}_2) + p(\mathcal{N}_3|\boldsymbol{x}) \times p(C_1|\mathcal{N}_3) \quad (5.64)$$

Cette dernière équation est facilement généralisable au cas de K composantes et C clusters par :

$$\forall i \in [\![1; C]\!] \quad p(C_i|\boldsymbol{x}) = \sum_{j=1}^{K} p(\mathcal{N}_j|\boldsymbol{x}) \times p(C_i|\mathcal{N}_j) \quad (5.65)$$

Après expérimentations, la moyenne obtenue sur l'ensemble des jeux de données est égale à 0,75, contre 0,86 pour la version originale de GSC, ce qui correspond à une baisse de 11%. On peut donc en conclure que l'approche proposée ici n'est pas bénéfique, du moins sur les données testées.

Ces résultats paraissent toutefois curieux. En effet, l'idée de combiner les probabilités du mélange Gaussien à celles du partitionnement flou semblait être intéressante. Ces performances en demi-teinte, bien que difficiles à expliquer pour le moment, peuvent provenir du fait que trop peu de données (à savoir autant que le nombre de composantes du MMG optimal) sont disponibles pour appliquer l'algorithme FCM.

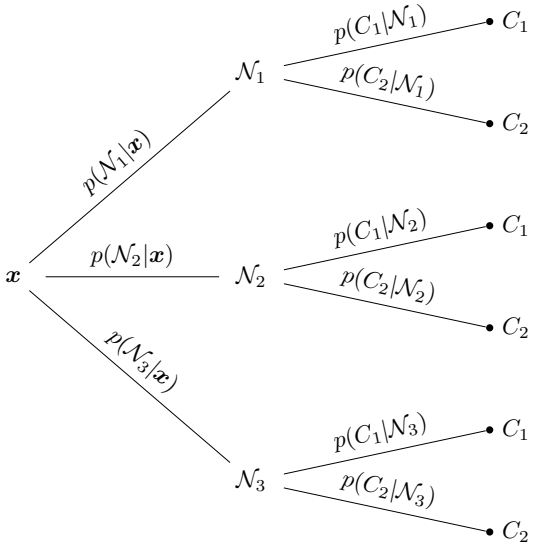


Figure 5.17. Arbre de probabilité avec une donnée $\boldsymbol{x}$, trois composantes Gaussiennes $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ et deux clusters C_1, C_2.

5.4.7 Conclusion

Nous avons donc proposé dans cette section un algorithme de partitionnement de données par distribution qui améliore le clustering par MG traditionnel. Ce dernier, nommé Gaussian Spectral Clustering (GSC), combine la modélisation par mélange Gaussien et le clustering spectral afin de fusionner les distributions normales similaires. La méthode détaillée est capable de découvrir des clusters de forme diverse, contrairement au clustering par MG classique qui est limité aux groupes elliptiques. Deux variantes ont été développées : la première pour laquelle le nombre de clusters à trouver est donné en entrée, la deuxième totalement non-paramétrique. De plus, il est possible d'ajouter des contraintes, aussi bien sur le nombre de composantes du modèle que sur celui de clusters, afin d'accélérer le processus et d'obtenir des résultats réalistes vis-à-vis de l'application considérée. Les expériences préliminaires menées montrent des performances plutôt encourageantes. Ce travail a donné lieu à une publication en conférence internationale :

> J. Muzeau, M. Oliver-Parera, P. Ladret et P. Bertolino, "Combining Mixture Models and Spectral Clustering for Data Partitioning", International Conference on Image Analysis and Recognition, 2020. [Muz+20]

Plusieurs perspectives sont à considérer pour la suite. Le premier vise à réaliser davantage d'expérimentations, afin de permettre une comparaison avec d'autres

algorithmes de clustering, également sur d'autres jeux de données, notamment en plus grande dimension. Deuxièmement, les pistes d'amélioration suggérées en sous-section 5.4.6 sont à creuser plus en profondeur. De plus, il pourrait être intéressant d'essayer d'autres variantes de l'algorithme EM, surtout en ce qui concerne l'initialisation du modèle. Enfin, les itérations autour de K-means étant sources d'erreurs, une solution différente pourrait être envisagée, particulièrement une méthode permettant de déterminer le nombre de clusters de manière immédiate, sans itération.

5.5 Conclusion

Ce chapitre a permis d'introduire la problématique générale du partitionnement de données, ou *clustering*, et son intérêt dans le contexte du projet MIVAO. Un état de l'art des techniques de ce milieu a été présenté. Ensuite, nous avons proposé deux algorithmes de partitionnement : le premier, EN-DBSCAN, est une généralisation de la méthode originale DBSCAN. Le deuxième, nommé GSC, concilie les clusterings par mélange Gaussien et spectral. Les résultats sont satisfaisants mais des pistes intéressantes d'amélioration s'offrent à nous.

En ce qui concerne l'objectif de comptage des passagers sur les télésièges, nous avons vu dans le chapitre précédent comment faire apparaître cette information à partir des images originales via la réduction de dimension. Nous venons de comprendre comment il est possible de partitionner des données en différents groupes. Nous combinerons dans le chapitre suivant ces deux aspects afin d'atteindre le but initial.

Evaluation qualitative

6

Sommaire

6.1 Introduction

Nous présentons dans ce chapitre les résultats obtenus par le pipeline proposé en chapitre 3 vis-à-vis du comptage des passagers présents sur n'importe quelle remontée mécanique, et ce de la manière la plus non-supervisée possible. Nous rappelons que ce dernier fonctionne comme suit :

1. D'abord, une réduction de dimension est appliquée sur les images unitaires afin de ne conserver que l'information utile.
2. Ensuite, les passages similaires (*i.e.* ceux dont les véhicules transportent le même nombre de personnes) sont regroupés via partitionnement de données.
3. Finalement, une donnée par groupe déterminé lors de l'étape précédente est annotée manuellement dans le but d'obtenir le véritable nombre de passagers de chaque véhicule.

La suite de ce chapitre est organisée comme suit. Nous détaillons en section 6.2 chaque étape du pipeline en profondeur. La section 6.3 est dédiée aux résultats obtenus. Enfin, une conclusion et des perspectives sont apportées en section 6.4.

6.2 Compléments algorithmiques

Nous détaillons en profondeur dans cette section les différentes étapes du pipeline proposé.

6.2.1 Données en entrée

Commençons par évoquer les données dont nous disposons. Il s'agit d'images unitaires, c'est-à-dire de vignettes extraites des images issues de la caméra, centrées sur le véhicule à chaque instant, une fois ce dernier circulant à l'intérieur de la zone de détection (voir Figure 1.1). Le processus est appliqué sur les images d'une même vidéo afin de prévenir les effets néfastes des conditions météorologiques. Une vidéo contenant une dizaine de passages et une trentaine d'images unitaires pour chacun, environ 400 images unitaires sont récupérées par vidéo : ce nombre varie d'une remontée mécanique à l'autre. Pour rappel, un descriptif plus détaillé est donné en section 3.2.

6.2.2 Pré-traitements

Les images en entrée sont d'abord redimensionnées afin qu'elles présentent toutes la même taille. Après expérimentations, nous avons choisi empiriquement une taille de 50 pixels de haut par 50 pixels de large. Les images sont ensuite converties en niveaux de gris, étant donné que la couleur n'aide pas particulièrement à la compréhension de la scène (voir justification en section 2.3).

De plus, dans le cadre de l'application présentée ici, seule l'information apportée par les passagers de chaque véhicule est intéressante. Plus précisément, nous souhaitons supprimer du mieux que l'on peut les pixels appartenant à l'arrière-plan et qui constituent donc du bruit. Ainsi, nous appliquons aux images unitaires une fenêtre d'apodisation. Cette dernière est de type Gaussien et centrée sur le milieu de l'image. Quant à l'écart-type de cette fenêtre, il reste le même pour toutes les images d'une même remontée mécanique, il est toutefois fixé manuellement. Nous donnons en Figure 6.1 deux exemples d'images unitaires avant et après application de la fenêtre d'apodisation en question. On observe effectivement que le contenu au centre de l'image reste intact alors que la valeur des pixels au niveau des bords (notamment dans les coins) tend vers zéro. Cela permet en particulier de réduire l'effet de ces pixels lors de comparaisons futures entre images.

(a) Images unitaires originales.

(b) Images unitaires après application d'une fenêtre d'apodisation.

Figure 6.1. Exemples d'images unitaires avant et après application d'une fenêtre d'apodisation Gaussienne.

Enfin, les niveaux de gris sont normalisés dans l'intervalle $[0; 1]$, suite à une division par 255.

6.2.3 Réduction de dimension

Les Cartes de Diffusion (CD), telles que détaillées en sous-section 4.2.2, sont appliquées aux images précédentes, aplaties sous forme de vecteurs de taille $50 \times 50 = 2500$.

Evoquons d'abord la distance utilisée afin de comparer les images unitaires entre elles. Notre choix se porte sur l'IMage Euclidean Distance (IMED) [Wan+05]. Considérons deux images $\boldsymbol{x}$ et $\boldsymbol{y}$ de taille $h \times l$, mises sous forme de vecteurs par concaténation des lignes de pixels : ainsi, $\boldsymbol{x} = (x_1, \ldots, x_{hl})$ et $\boldsymbol{y} = (y_1, \ldots, y_{hl})$. La distance Euclidienne usuelle se calcule comme suit :

$$d_E^2(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{h \times l} (x_i - y_i)^2. \tag{6.1}$$

Quant à elle, l'IMED est définie par l'équation suivante :

$$d_{IME}^2(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2\pi} \sum_{i,j=1}^{h \times l} \exp\left(-\frac{\|\boldsymbol{P_i} - \boldsymbol{P_j}\|_2^2}{2}\right)(x_i - y_i)(x_j - y_j), \tag{6.2}$$

où $\boldsymbol{P_i}, \boldsymbol{P_j}$ représentent les coordonnées des $i^{\text{ème}}$ et $j^{\text{ème}}$ pixels. La principale différence avec la distance Euclidienne traditionnelle est l'inclusion d'une information de distance spatiale entre chaque pixel, représentée par la fonction Gaussienne, en plus de celle entre niveaux de gris. Les liens spatiaux entre pixels sont ainsi pris en compte, ce qui permet une robustesse aux faibles perturbations présentes dans l'image, contrairement à la distance usuelle. Une version accélérée de l'IMED est proposée dans [SF08].

La deuxième étape des CD vise à déterminer la similarité entre chaque paire d'images. Cette similarité $s_{\boldsymbol{xy}}$ est calculée via la distance $d_{IME}(\boldsymbol{x}, \boldsymbol{y})$ entre deux données $\boldsymbol{x}$ et $\boldsymbol{y}$ par :

$$s_{\boldsymbol{xy}} = \exp\left(-\frac{d_{IME}^2(\boldsymbol{x}, \boldsymbol{y})}{2\sigma^2}\right). \tag{6.3}$$

Une heuristique usuelle permet de fixer le paramètre σ comme étant la **médiane des distances paires à paires**, autrement dit

$$\sigma = \text{médiane}\ \{d_{IME}(\boldsymbol{x}, \boldsymbol{y}) \quad \forall (\boldsymbol{x}, \boldsymbol{y})\}. \tag{6.4}$$

Un autre paramètre de ces cartes de diffusion est la dimension d des données réduites. Comme mentionné dans [Tal+13], elle peut être déduite de l'analyse spectrale de la

matrice Markovienne des données. Pour ce faire, considérons l'exemple d'une vidéo d'une remontée mécanique et appliquons l'algorithme des cartes de diffusion sur les images unitaires. Les premières valeurs propres de la matrice Markovienne obtenue sont données, dans l'ordre décroissant, en Figure 6.2. Notons que la première valeur propre est toujours égale à 1, comme expliqué en sous-section 4.2.2, elle est donc retirée du graphique. On observe un écart significatif entre la deuxième et la troisième valeur propre : c'est ce que l'on appelle le **fossé spectral**. Ainsi, les deux premières valeurs propres contiennent plus d'informations que les autres, la dimension réduite d peut par conséquent être choisie comme étant égale à deux. De manière pratique, la dimension appropriée est obtenue via le maximum de la dérivée numérique.

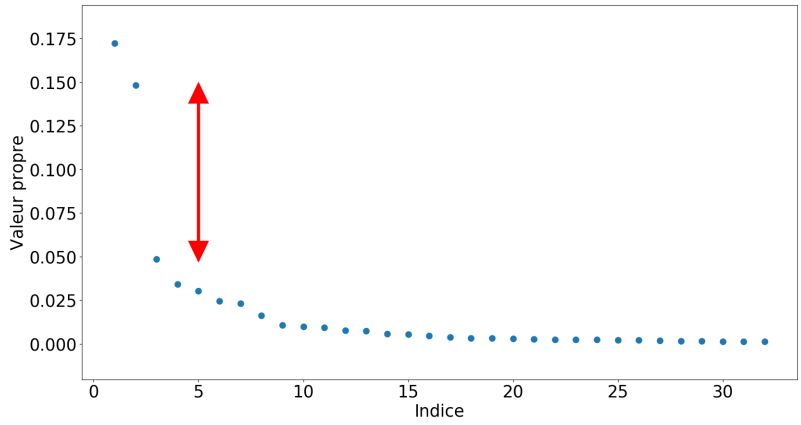


Figure 6.2. Valeurs propres triées dans l'ordre décroissant. On observe un fossé (flèche rouge) entre la deuxième et la troisième valeur propre. Ainsi, une réduction à deux dimensions paraît suffisante.

Précisons que la valeur précédente n'est valable que pour cet exemple. En effet, comme on peut le voir grâce à l'histogramme présenté en Figure 6.3, la dimension réduite évolue, sur l'ensemble de la base de données, entre 2 et 16, en fonction du contenu de chaque vidéo. Nous pouvons nous interroger quant aux faibles valeurs obtenues. Comme expliqué en sous-section 4.2.2, l'algorithme des CD est capable de trouver les paramètres sous-jacents qui ont permis de générer les données analysées. Dans notre cas, les variables latentes qui évoluent tout au long de la scène sont peu nombreuses, car nous avons fait en sorte que seules les informations liées aux passagers se démarquent. Il est donc logique d'obtenir une dimension réduite faible, en moyenne.

Un dernier élément que nous souhaitons souligner est le suivant. Il est important d'avoir un nombre conséquent d'images à traiter, pour chaque vidéo, afin de pouvoir en tirer des conclusions vraiment significatives. En effet, plus on a d'images, plus il est

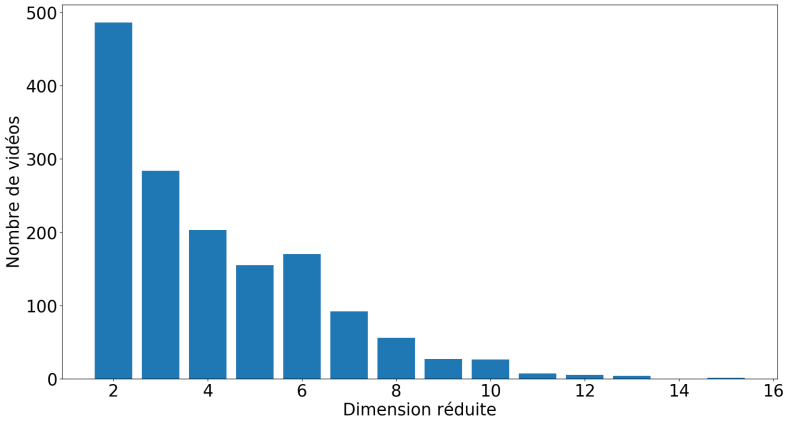


Figure 6.3. Histogramme des dimensions réduites obtenues grâce aux cartes de diffusion pour l'ensemble des vidéos du projet MIVAO.

facile pour les cartes de diffusion de déterminer la variété mathématique sur laquelle les données reposent. Un problème survient par exemple lorsqu'une vidéo est plus courte que les autres, certains passages manquent et le phénomène est amplifié pour les images unitaires (étant donné qu'un passage correspond en moyenne à 40 vignettes). Pour contrecarrer cette difficulté, nous proposons d'ajouter aux images unitaires originales leur double additionné d'un **bruit Gaussien** de faible variance (typiquement 0,01). De cette manière, on garantit un nombre suffisant d'images pour appliquer l'algorithme des cartes de diffusion. Il est également important de noter que les images bruitées n'apportent aucune information qui n'est pas déjà présente dans les données originales, du fait de leur forte similarité avec les images unitaires de base. Ce genre d'astuces porte le nom d'**augmentation de données** [Kri+12 ; Fad+17 ; Sze+15].

6.2.4 Partitionnement de données

Suite à la réduction de dimension réalisée par les cartes de diffusion, comme expliqué précédemment, nous souhaitons regrouper les passages de véhicules transportant le même nombre de personnes. Pour ce faire, nous appliquons une technique de partitionnement de données (ou *clustering*), et plus particulièrement EN-DBSCAN présentée en section 5.3. Comme on a pu le voir sur quelques exemples (*cf.* Figure 4.4), les cartes de diffusion aboutissent parfois à des groupes qui peuvent se chevaucher localement ou qui présentent une "cassure". C'est pourquoi la prise en compte de la distribution locale via les voisinages ellipsoïdaux est cruciale, d'où le choix de cette méthode.

Les données générées par les cartes de diffusion sont normalisées, c'est-à-dire centrées par rapport à la moyenne et réduites par rapport à l'écart-type de chaque dimension. Par la suite, différents partitionnements sont calculés avec le paramètre ε compris dans la plage $]0;1]$, avec un pas égal à $0,05$. En ce qui concerne le nombre minimal $MinPts$ de points qu'un voisinage doit contenir afin de pouvoir être le commencement d'un nouveau cluster, nous le choisissons égal à quatre. Cette décision provient du fait que nous connaissons les données et ce qu'elles représentent. Nous savons qu'un passage contient une quarantaine d'images unitaires et qu'avoir affaire à quatres images *outliers* consécutives est peu probable, d'où notre choix. Ce genre de situations peut par exemple survenir lors d'un suivi défaillant de véhicule : dans ce cas, les images unitaires obtenues ne sont pas centrées sur le siège et la distance avec les autres images est grande, ce qui donne lieu à des points isolés après réduction de dimension.

6.2.5 Propagation d'étiquettes

Choix des données à annoter manuellement

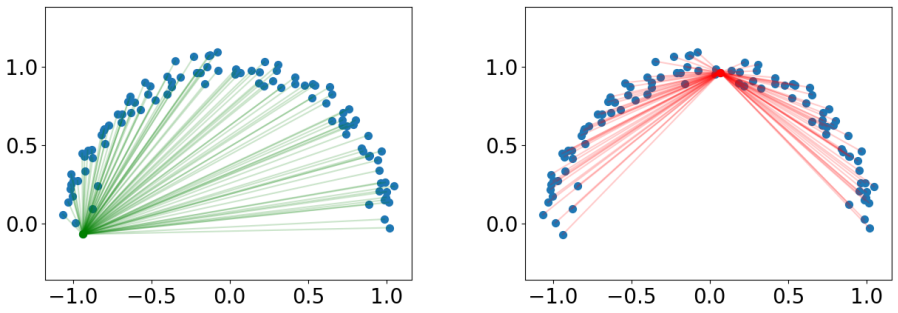
Le résultat rendu à l'étape précédente est un regroupement des passages similaires d'une vidéo. Toutefois, le nombre exact de passagers transportés n'entre pas en jeu. Pour ce faire, nous proposons d'inclure cette information via l'annotation manuelle d'une image par cluster déterminé. La donnée annotée est choisie comme étant celle qui possède le plus de voisins. De cette manière, nous garantissons que l'image choisie ne soit pas *outlier* et soit effectivement représentative du cluster auquel elle appartient.

Pour ce faire, nous procédons comme suit. Considérons un jeu de données $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ partitionné en K groupes, à savoir $C_1, \ldots, C_K$. Pour chaque point $\boldsymbol{x_i}$, nous calculons la Somme des Distances Euclidiennes Quadratiques (SDEQ) entre $\boldsymbol{x_i}$ et chaque autre point appartenant au même cluster. Autrement dit, on détermine :

$$\forall i \in [\![1;n]\!] \quad SDEQ_i = \sum_{\boldsymbol{x_j} \in C_{k_i}} \|\boldsymbol{x_j} - \boldsymbol{x_i}\|_2^2, \tag{6.5}$$

où k_i représente l'indice du cluster auquel la donnée i appartient. Ensuite, pour chaque cluster, le point dont la SDEQ est la plus faible est considéré comme celui entouré du plus de voisins et est donc choisi pour l'annotation manuelle. Naturellement, nous associons la même étiquette aux images appartenant au même passage que la donnée annotée.

Afin d'illustrer ce choix, la Figure 6.4 montre un cluster (points bleus) et deux possibilités pour la donnée à annoter manuellement. Le calcul de la SDEQ pour le point vert donne 196,6 (figure de gauche), alors qu'elle vaut 72,4 pour le point rouge (figure de droite) : le choix n°2 est donc plus intéressant.



(a) Choix 1, point vert. **(b)** Choix 2, point rouge.

Figure 6.4. Illustration du choix de la donnée à annoter grâce à la SDEQ.

Transduction probabiliste

Une fois certaines données annotées manuellement, il est nécessaire de propager ces étiquettes aux autres images : on parle alors de **transduction**. La solution naïve consiste à considérer que toutes les données d'un même cluster possèdent la même étiquette, c'est-à-dire que le même nombre de passagers est assigné à tous les points. Toutefois, cette méthode peut être fortement altérée par les erreurs éventuelles créées par le partitionnement de données (*e.g.* si deux véritables clusters sont considérés comme un seul et unique).

Afin de garantir une certaine robustesse, nous proposons de plutôt utiliser la technique de semi-supervision présentée par Zhou *et al.* [Zho+04]. Cette dernière consiste sommairement à propager l'information portée par les annotations manuelles de manière probabiliste de proche en proche. Concrètement, cela signifie qu'une donnée non-annotée est plus affectée par une étiquette utilisée pour l'annotation d'un point voisin que par celle d'un point éloigné.

Considérons le cas général de K classes et d'un jeu de données $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ dont on connaît la classe d'appartenance de certains points, pas les autres.

1. Consuitre la matrice d'affinité $\boldsymbol{W} = (W_{ij})$, de taille $n \times n$, par :

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2}{2\sigma^2}\right) & \text{si } i \neq k, \\ 0 & \text{sinon.} \end{cases} \tag{6.6}$$

2. Créer la matrice $\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, où $\boldsymbol{D}$ représente la matrice diagonale des sommes ligne par ligne de celle d'affinité.
3. Initialiser $\boldsymbol{F}(t=0) = \boldsymbol{Y} = (y_{ij})$, avec

$$\forall (i,j) \in [\![1;n]\!] \times [\![1;K]\!] \quad y_{ij} = \begin{cases} 1 & \text{si } \boldsymbol{x_i} \text{ appartient à la classe } j, \\ 0 & \text{sinon.} \end{cases} \tag{6.7}$$

4. Itérer $\boldsymbol{F}(t=k+1) = \alpha \boldsymbol{S}\boldsymbol{F}(t=k) + (1-\alpha)\boldsymbol{Y}$ jusqu'à convergence vers $\boldsymbol{F}^*$.
5. Chaque valeur finale $\boldsymbol{F}^*_{ij}$ correspond à la probabilité que la donnée i appartienne à la classe j. On peut donc étiquetter chaque point en conséquence.

Le paramètre $\alpha \in]0;1[$ traduit la vitesse de propagation de l'information d'appartenance à une classe spécifique des données étiquetées au début de l'algorithme vers les différents voisins. Notons que cette méthode peut être vue comme une amélioration de [ZG02] grâce à l'ajout de régularisation, caractérisée par α.

Agrégation via l'information d'appartenance à chaque passage

Suite à l'application de cet algorithme, chacune des images originales possède une étiquette représentant le nombre de passagers présents sur le véhicule associé. Nous pouvons par la suite utiliser une information complémentaire, qui consiste à dire que toutes les images d'un même passage doivent être étiquetées de la même manière. Comme expliqué précédemment, la matrice $\boldsymbol{F}^*$ obtenue associe à chaque image une probabilité d'appartenir à telle ou telle classe. Afin de déterminer la même information mais pour chaque passage, nous utilisons le lien suivant :

$$\mathbb{P}\left(j \text{ passagers} \mid \text{passage } i\right) = \prod_{\text{image } k \,\in\, \text{passage } i} \mathbb{P}\left(j \text{ passagers} \mid \text{image } k\right). \tag{6.8}$$

En pratique, nous passons plutôt au logarithme népérien, c'est-à-dire :

$$\ln \mathbb{P}\left(j \text{ passagers} \mid \text{passage } i\right) = \sum_{\text{image } k \,\in\, \text{passage } i} \ln \mathbb{P}\left(j \text{ passagers} \mid \text{image } k\right), \tag{6.9}$$

afin d'éviter des erreurs de dépassement liées à une multiplication entre valeurs proches de zéro. Par la suite, la classe dont la probabilité est la plus élevée est associée au passage en question.

Métriques de classification utilisées

Finalement, les résultats obtenus suite à la transduction sont équivalents à ceux que l'on peut avoir après classification, c'est-à-dire une classe d'appartenance (liée au nombre de passagers présents) pour chaque image ou passage. Ainsi, les mesures usuelles peuvent être utilisées ici, à savoir PPC, Rappel et F-mesure. Rappelons que nous faisons référence ici à des métriques **multi-classes**.

Afin de situer les différents scores à l'oeuvre ici, considérons la sortie de notre pipeline sur un exemple simple. La vidéo concernée contient seize passages comprenant zéro, trois ou quatre passagers. La matrice de confusion associée est donnée en Tableau 6.1. On peut comprendre à partir de ces résultats que sept passages ont été correctement identifiés comme ne comprenant aucun passager et qu'un passage a été prédit avec zéro passager alors qu'il en comprenait en réalité trois. On peut en déduire les PPC, le Rappel et la F-mesure pour chaque classe de la vérité-terrain (voir Tableau 6.2). Les moyennes arithmétiques pour chaque mesure sont dites **macro**. On peut également pondérer les scores par le nombre d'éléments de chaque classe, c'est-à-dire sept, cinq et quatre passages respectivement. Enfin, la moyenne dite **micro** est déterminée en divisant le nombre de passages correctement étiquetés par le nombre total d'éléments, autrement dit $(7 + 2 + 3)/16 = 0,75$.

	Prédiction		
	0	3	4
0 passager	7	0	0
3 passagers	1	2	2
4 passagers	0	1	3
Vérité-terrain			

Table 6.1. Matrice de confusion sur un exemple simple.

	PPC	Rappel	F-mesure
0 passager	0,875	1,000	0,933
3 passagers	0,667	0,400	0,500
4 passagers	0,600	0,750	0,667
Moyenne micro			0,750
Moyenne macro	0,714	0,717	0,700
Moyenne pondérée	0,741	0,750	0,731

Table 6.2. Différentes mesures calculées suite à la classification multi-étiquettes dont la matrice de confusion est donnée en Tableau 6.1.

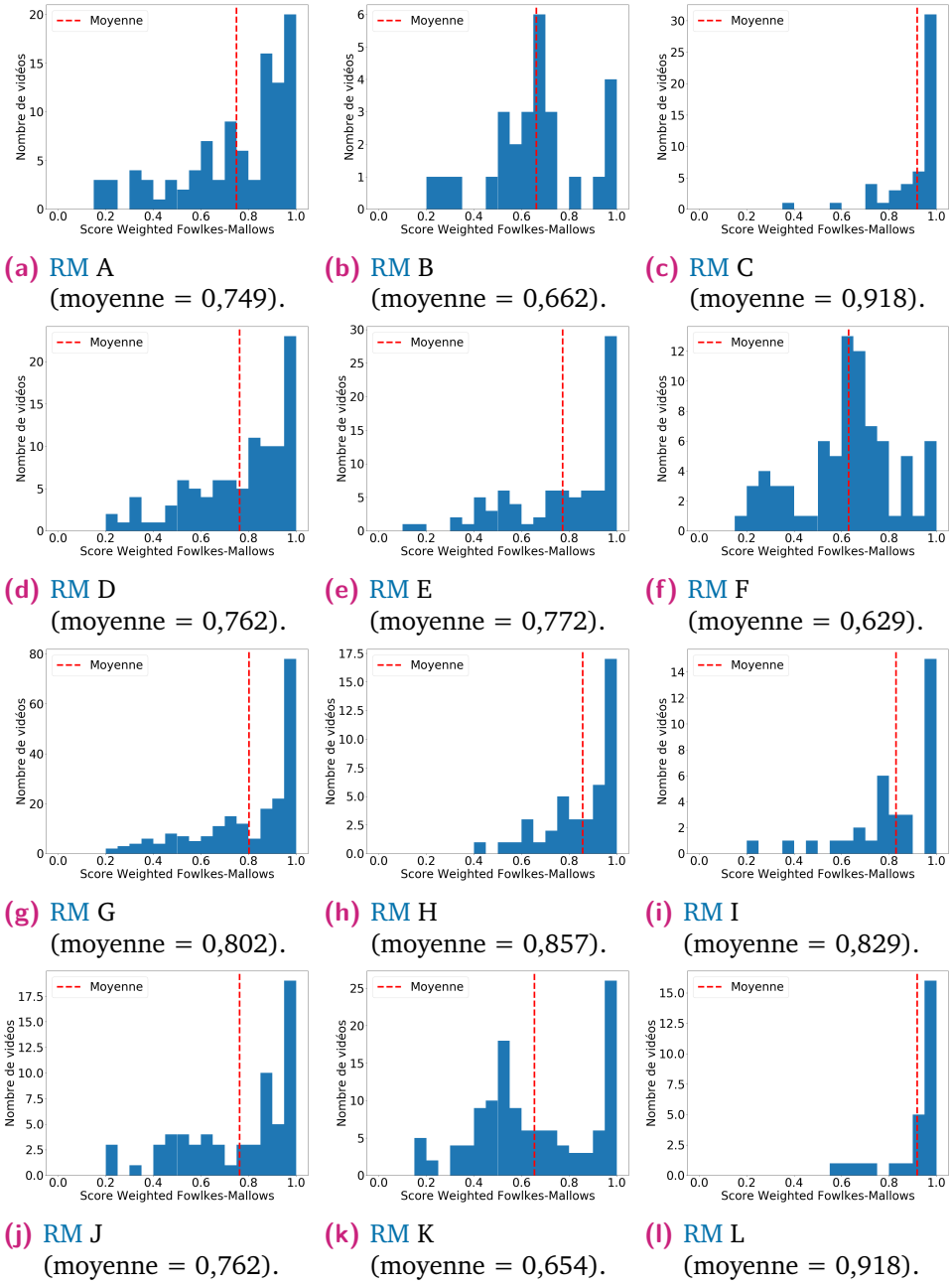
6.3 Résultats

Les résultats présentés ici sont séparés en deux parties : ceux obtenus après **(i)** clustering, et **(ii)** transduction.

Dans un premier temps, les résultats du partitionnement de données sont fournis en Figure 6.5. Pour ce faire, on présente les histogrammes (sur 20 *bins*) des scores Weighted Fowlkes-Mallows (WFM), pour les 19 remontées mécaniques concernées. Pour rappel, ce dernier représente le score de Fowlkes-Mallows [FM83] pondéré par le pourcentage d'aire couverte, afin de prendre en compte les données considérées comme du bruit (*cf.* section 5.3.8 pour plus d'informations). La moyenne est représentée par une ligne verticale rouge, elle est également donnée en légende pour chaque télésiège.

De manière générale, on peut conclure que ce pipeline donne de bonnes performances de partitionnement : en effet, onze RM parmi les 19 au départ présentent une moyenne supérieure à 80%. Cela signifie que le processus proposé est capable, en moyenne, de regrouper les images montrant le même nombre de passages dans 80% des cas. On peut noter les excellents résultats obtenus pour les RM C, I, O, P et S (Figures 6.5c, 6.5i, 6.5o, 6.5p et 6.5s). En revanche, plus de difficultés se posent sur les RM A, B, F et K (Figures 6.5a, 6.5b, 6.5f et 6.5k).

Dans l'immense majorité des cas, les mauvais résultats de partitionnement de données sont liés à l'échec des cartes de diffusion. Deux exemples en trois dimensions, afin de permettre une visualisation aisée, sont affichés en Figure 6.6. On peut clairement noter que la distinction entre les différents types de passages est



(a) RM A (moyenne = 0,749). (b) RM B (moyenne = 0,662). (c) RM C (moyenne = 0,918).

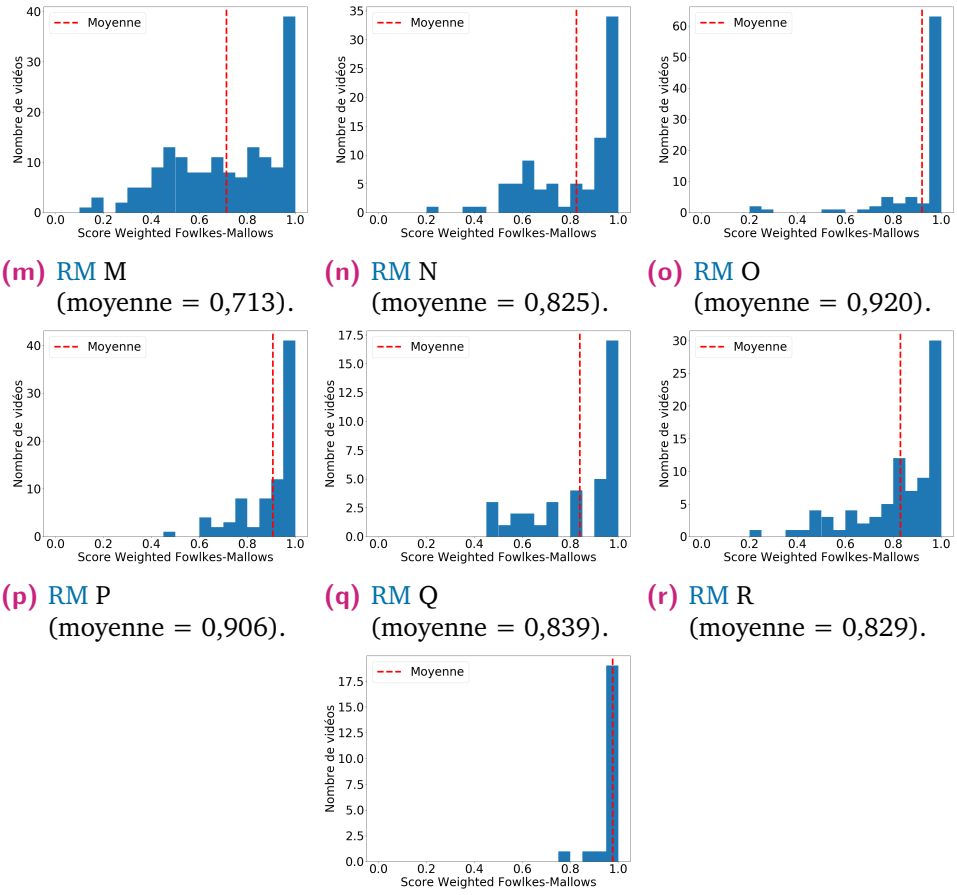
(d) RM D (moyenne = 0,762). (e) RM E (moyenne = 0,772). (f) RM F (moyenne = 0,629).

(g) RM G (moyenne = 0,802). (h) RM H (moyenne = 0,857). (i) RM I (moyenne = 0,829).

(j) RM J (moyenne = 0,762). (k) RM K (moyenne = 0,654). (l) RM L (moyenne = 0,918).

Figure 6.5. Histogrammes des scores Weighted Fowlkes-Mallows après partitionnement, remontée par remontée.

quasi-impossible à l'oeil nu. Même si un ordre apparaît visuellement (grâce à la vérité-terrain), l'algorithme EN-DBSCAN est incapable de le récupérer de manière automatique, car il fonctionne en distinguant les zones de haute densité de celles de faible densité. Dans ces cas-là, un seul cluster est découvert par la méthode de clustering.



(m) RM M (moyenne = 0,713).

(n) RM N (moyenne = 0,825).

(o) RM O (moyenne = 0,920).

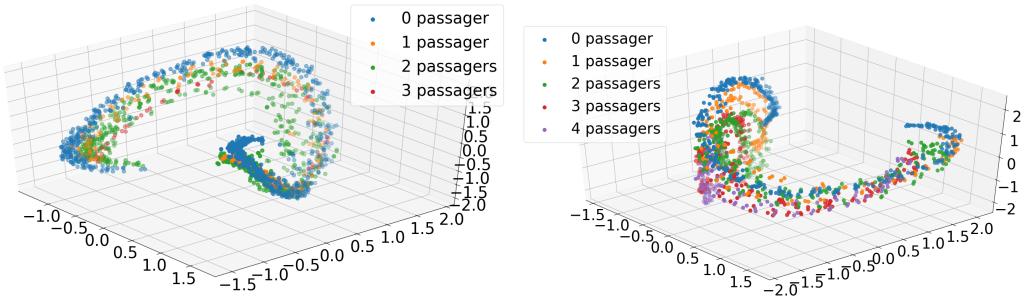
(p) RM P (moyenne = 0,906).

(q) RM Q (moyenne = 0,839).

(r) RM R (moyenne = 0,829).

(s) RM S (moyenne = 0,976).

Figure 6.5. Histogrammes des scores Weighted Fowlkes-Mallows après partitionnement, remontée par remontée (suite).



(a) Exemple 1.

(b) Exemple 2.

Figure 6.6. Exemples pour lesquels les cartes de diffusion échouent. Chaque couleur représente un nombre de passagers différent, obtenu grâce à la vérité-terrain.

Concrètement, on peut expliquer ces mauvaises performances par le fait que la distance utilisée afin de comparer les images entre elles (IMED plus précisément)

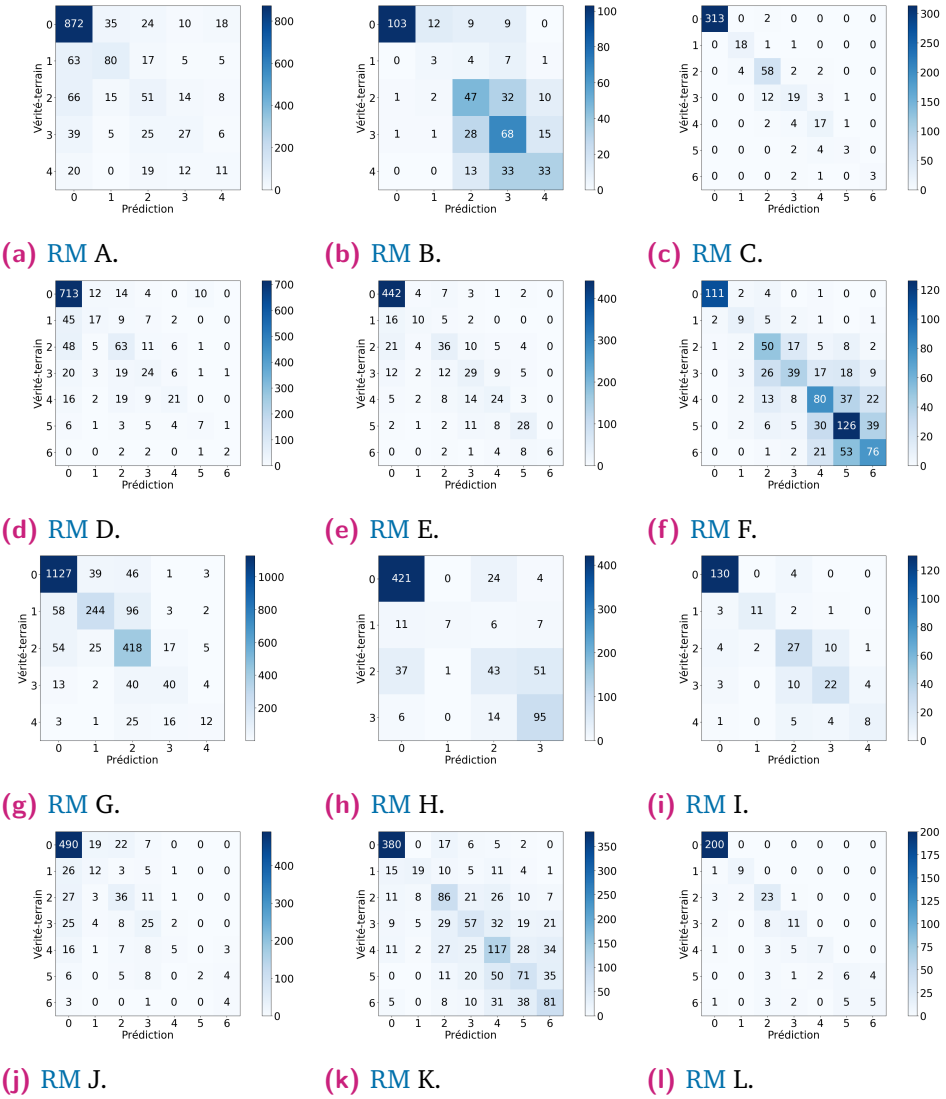
n'est pas optimale et ne permet pas de distinguer correctement les différents types de passages. Des exemples d'images unitaires sont donnés en Figure 6.7. On peut remarquer que, dans ces cas, l'arrière-plan est loin d'être uniforme. De fait, malgré le fenêtrage Gaussien évoqué en sous-section 6.2.2, il participe au calcul et engendre des distances entre images faussées qui ne permettent pas l'extraction correcte de l'information du nombre de passagers via les cartes de diffusion.

Figure 6.7. Deux situations dans lesquelles l'IMED échoue à retranscrire les réelles différences. Pour ces deux RM différentes, les distances paires à paires calculées sont troublées par l'intervention de l'arrière-plan.

Toutefois, comme expliqué dans la section précédente, ces valeurs ne sont pas représentatives du nombre exact de passagers pour chaque passage, d'où la dernière étape de transduction. Nous affichons en Figure 6.8 les matrices de confusion obtenues sur l'ensemble des vidéos de chaque remontée mécanique. Précisons que les valeurs données concernent les passages et non pas les images.

Nous présentons en Figure 6.9 les histogrammes des moyennes micro, remontée par remontée. La moyenne est également représentée par une ligne verticale rouge et répétée en légende pour chaque télésiège. Nous choisissons d'afficher les moyennes micro (voir Tableau 6.2) car elles résument en une seule valeur la qualité de la classification.

De même que précédemment, les résultats sont encourageants. Ils suivent la même tendance que ceux obtenus après partitionnement (voir Figure 6.5), c'est-à-dire que les remontées mécaniques qui aboutissent à une moyenne des scores WFM élevée aboutissent également à une moyenne des scores micro élevée, et vice-versa. Il est toutefois à noter que nous avons affaire ici à une classification multi-étiquettes, il est donc plus difficile d'obtenir des scores proches de 1. De plus, pour chaque vidéo, le nombre de passages étant assez faible, une erreur de classification engendre une forte diminution en termes de moyenne micro (par exemple, 10% dans le cas d'une vidéo de dix passages).



(a) RM A. (b) RM B. (c) RM C.

(d) RM D. (e) RM E. (f) RM F.

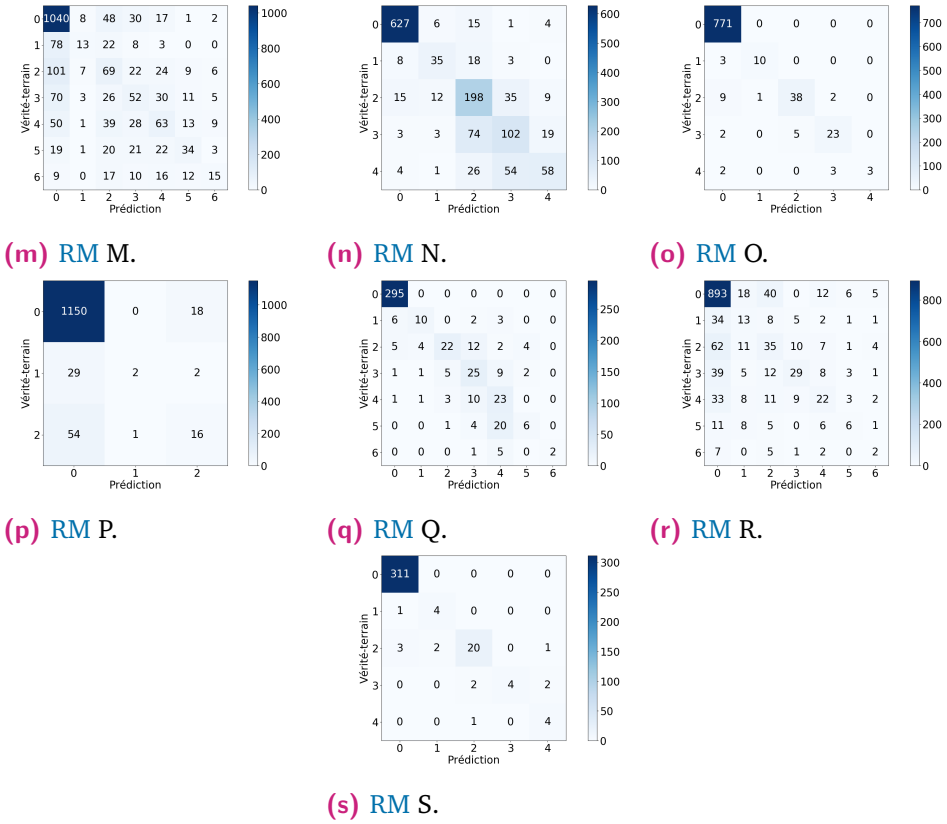
(g) RM G. (h) RM H. (i) RM I.

(j) RM J. (k) RM K. (l) RM L.

Figure 6.8. Matrices de confusion en termes de passages après transduction, remontée par remontée.

On peut de plus souligner l'apport bénéfique de la gestion des erreurs de partitionnement de données via l'algorithme proposé dans [Zho+04]. La moyenne micro globale sur l'ensemble des remontées mécaniques est ici égale à 0,754. Si l'on considère l'approche naïve, qui consiste à étiquetter l'ensemble des données d'un cluster de la même manière, alors on obtient une moyenne globale égale à 0,577, ce qui correspond à une diminution de plus de 20%. Ainsi, la méthode [Zho+04] permet de corriger des erreurs générées par l'algorithme EN-DBSCAN et d'aboutir à une classification plus proche de la réalité.

Une dernière observation intéressante que l'on peut tirer de ces résultats est la suivante. Nous pouvons nous ramener au contexte de la Partie I en associant les cas



(m) RM M. (n) RM N. (o) RM O.

(p) RM P. (q) RM Q. (r) RM R.

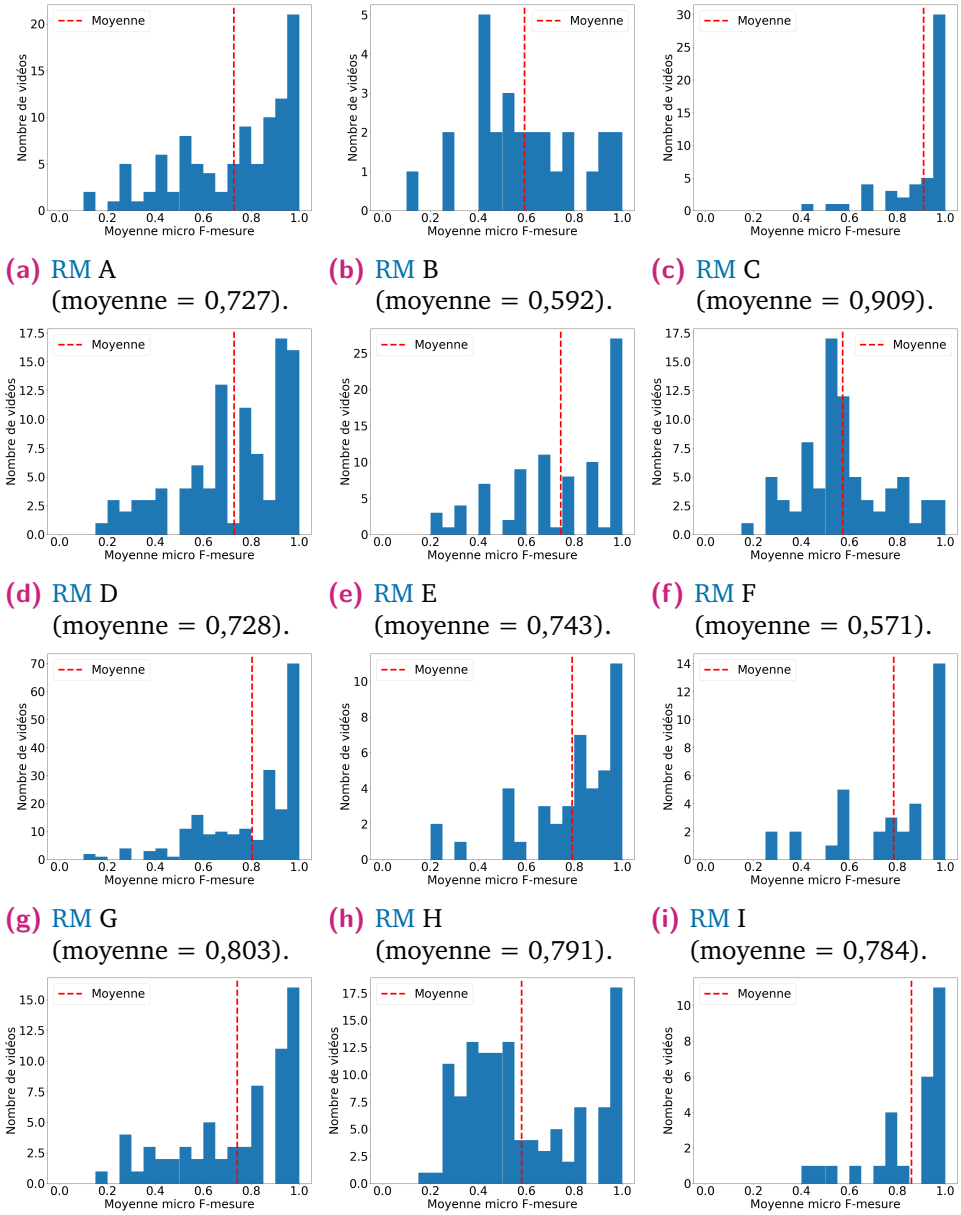
(s) RM S.

Figure 6.8. Matrices de confusion en termes de passages après transduction, remontée par remontée (suite).

où le nombre de passagers est supérieur ou égal à 1 à la classe **non-vide**. La F-mesure moyenne alors obtenue sur l'ensemble des remontées mécaniques est égale à 0,907. Cette valeur est à mettre en comparaison avec celles déterminées au chapitre 2. Pour rappel, nous avions déduit, dans un premier temps, une F-mesure égale à 0,867 en combinant les analyses discriminantes linéaire et quadratique, 0,958 après inclusion d'informations *a priori* (sous la forme de boîtes englobantes au niveau des torses et des jambes des passagers potentiels). Cette remarque est intéressante car les résultats obtenus par le pipeline proposé sont compétitifs, tout en gardant à l'esprit que ce dernier est en majorité non-supervisé et ne nécessite aucune connaissance *a priori* sur la remontée mécanique considérée.

6.4 Conclusion

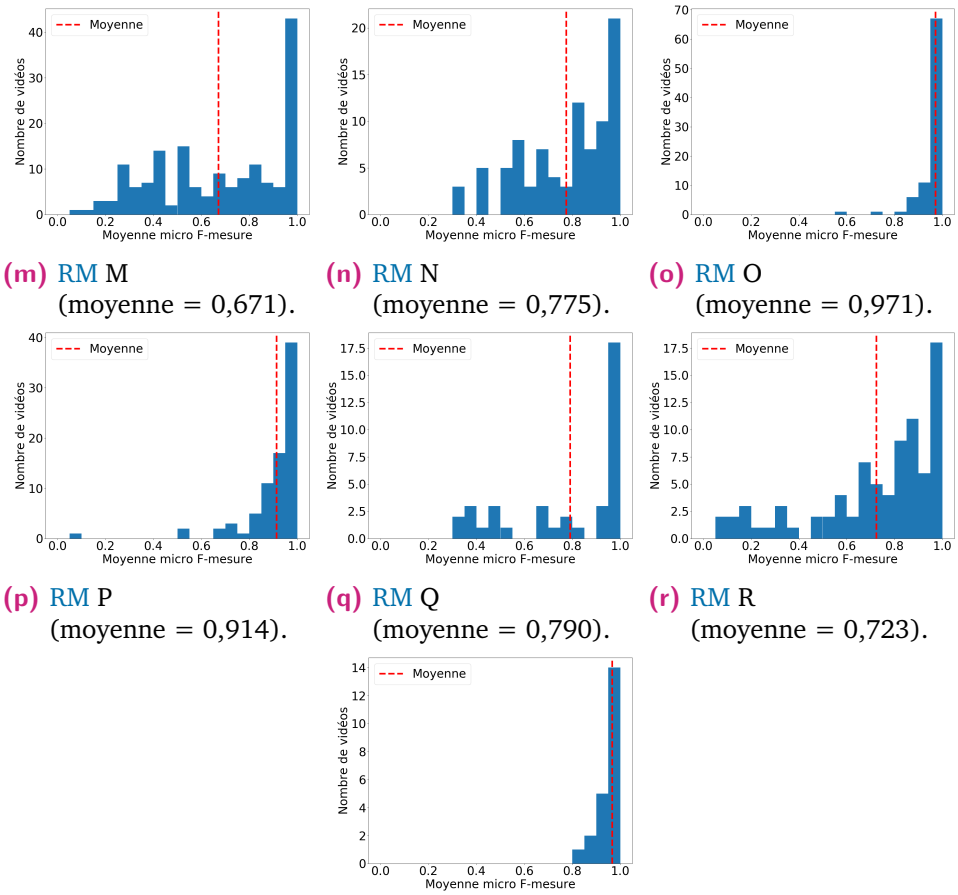
Dans cette section, nous avons détaillé chaque étape du pipeline proposé, dont le but est le comptage des passagers de chaque véhicule de la manière la moins



(a) RM A (moyenne = 0,727).
(b) RM B (moyenne = 0,592).
(c) RM C (moyenne = 0,909).
(d) RM D (moyenne = 0,728).
(e) RM E (moyenne = 0,743).
(f) RM F (moyenne = 0,571).
(g) RM G (moyenne = 0,803).
(h) RM H (moyenne = 0,791).
(i) RM I (moyenne = 0,784).
(j) RM J (moyenne = 0,741).
(k) RM K (moyenne = 0,580).
(l) RM L (moyenne = 0,858).

Figure 6.9. Histogrammes des moyennes micro après transduction, remontée par remontée.

supervisée possible. En l'occurence, il se compose d'une première étape de réduction de dimension, d'un partitionnement de données grâce à la méthode EN-DBSCAN proposée, puis d'un processus de transduction. Nous avons présenté des résultats expérimentaux qui montrent des performances de dénombrement de passagers satisfaisantes, aussi bien en termes de clustering qu'en matière de classification. En



(m) RM M (moyenne = 0,671).
(n) RM N (moyenne = 0,775).
(o) RM O (moyenne = 0,971).
(p) RM P (moyenne = 0,914).
(q) RM Q (moyenne = 0,790).
(r) RM R (moyenne = 0,723).
(s) RM S (moyenne = 0,966).

Figure 6.9. Histogrammes des moyennes micro après transduction, remontée par remontée (suite).

particulier, nous avons pu constater qu'une classification vide/non-vide fortement non-supervisée était possible et prometteuse.

Différentes perspectives sont imaginables en ce qui concerne ce travail. La première concerne la distance utilisée afin de comparer les différentes images unitaires. L'inclusion de l'information de couleurs pourrait être bénéfique étant donné que les combinaisons des skieurs sont souvent colorées, cela permettrait également de pouvoir différencier les skieurs de l'arrière-plan, terne. Il serait également intéressant d'investiguer d'autres choix pour la distance elle-même. De plus, en ce qui concerne le partitionnement de données, nous avons pu remarquer que la différence entre les types de passages était plus marquée à certains instants plutôt qu'à d'autres. La prise en compte de ce phénomène pourrait améliorer les performances. Enfin, l'apport de la localisation de chaque passager potentiel sur chaque véhicule, information constante pour une même remontée mécanique, pourrait être bénéfique : cette

information peut être amenée soit manuellement, soit automatiquement grâce au travail présenté en chapitre 7.

Troisième partie

Détection automatique de véhicules de remontée mécanique

Extraction des véhicules

7

Sommaire

7.1 Motivations

Nous nous intéressons dans cette partie à l'extraction des images unitaires, c'est-à-dire de vignettes centrées sur le véhicule d'intérêt une fois celui-ci entré dans la zone de détection, à partir des images issues de la caméra (voir Figure 1.1 pour rappel). Cette tâche peut se décomposer en deux sous-éléments : il est d'abord nécessaire de **détecter** le véhicule, puis de le **suivre** à travers les images.

En ce qui concerne la première étape, celle de détection, elle est réalisée dans le cadre du projet SIVAO par *template matching* [Bru09]. Concrètement, cela signifie qu'un modèle de véhicule est créé au préalable, manuellement lors de l'installation du système, puis est cherché dans l'image caméra, lors de l'exploitation, jusqu'à obtention d'un score de correspondance satisfaisant. Plus précisément, le *template matching* est réalisé ici entre les contours de l'image et ceux de l'objet à détecter. L'objet en question est le joint du véhicule, représenté en rouge sur la Figure 1.1b, à l'instant même où ce dernier entre dans la zone de détection, c'est-à-dire au niveau de la face du parallélépipède la plus proche de l'embarquement. Quant à l'étape de suivi, elle consiste simplement en une recherche de l'objet en question dans la zone approximative où il a été détecté à l'instant précédent. Précisons tout de même que ces étapes de détection et suivi se font à différents niveaux (grossiers puis fins) dans le but d'être robuste au bruit et d'obtenir une localisation précise dans l'image.

Bien que cette méthode soit intéressante en pratique, plusieurs problèmes se posent :

- Premièrement, le modèle évoqué précédemment est créé manuellement : on comprend aisément que cette tâche est chronophage et non-optimale.
- Deuxièmement, cela implique qu'un nouveau *template* est à élaborer à chaque installation du système de surveillance sur une nouvelle remontée mécanique, ce qui freine le déploiement de cette solution à grand échelle.
- Enfin, le *template* est indéformable, il n'évolue pas entre l'entrée et la sortie de la zone de détection. C'est une difficulté pour le suivi car la représentation visuelle du véhicule dans l'image se déforme au fil du temps, principalement à cause de la perspective.

Ainsi, l'idée que nous proposons à travers cette partie est la suivante : la modélisation de la trajectoire de la remontée mécanique dans le but de pouvoir localiser à chaque instant la position du véhicule dans l'image, et ce de manière automatique.

Ce chapitre est organisé comme suit : nous exposons en section 7.2 une méthode basée sur l'analyse fréquentielle afin de connaître automatiquement la durée d'un

passage. La section 7.3 est dédiée à la présentation d'une technique particulière de détection d'objets en mouvement que nous proposons et aux résultats obtenus. Enfin, une conclusion et des perspectives sont données en section 7.4.

7.2 Détermination de la durée d'un passage

Nous présentons dans cette section une méthode visant à obtenir la durée, en secondes ou de manière équivalente en nombre d'images, d'un passage pour une remontée mécanique particulière. Pour rappel, comme expliqué en section 1.2, la notion de passage est particulière au projet MIVAO et est définie comme étant l'ensemble des positions d'un véhicule à l'intérieur de la zone de détection. La méthode proposée ici émane de l'observation que, dans une vidéo de quelques minutes, la même scène se répète plusieurs fois, notamment en ce qui concerne la position des véhicules. L'évolution est périodique : ainsi, une analyse fréquentielle semble adaptée.

Précisons également que cette information n'est pas explicitement utilisée dans le projet SIVAO. Elle peut toutefois être utile à plusieurs égards. D'abord, en apportant des informations vis-à-vis de la physique du système, elle aide à la modélisation de la trajectoire d'un télésiège et donc à la construction de la zone de détection, et ce de manière **automatique**. Egalement, elle permet de savoir quand commencer à chercher le véhicule dans l'image. Enfin, on peut imaginer grâce à elle une remise à zéro de la détection et du suivi en cas d'erreurs.

On suppose ici travailler sur une vidéo d'une remontée mécanique en fonctionnement "normal", c'est-à-dire à vitesse constante et dans laquelle il n'y a aucun évènement extraordinaire (*e.g.* arrêt d'urgence). On peut également imaginer que cette vidéo est acquise lors de l'installation de la caméra, avant même tout autre processus.

Commencons par construire le tenseur $\boldsymbol{V}$ représentant les différentes images de la vidéo empilées les unes sur les autres de manière chronologique. Ainsi, si nous avons affaire à t images de h pixels de hauteur et l de largeur, alors le tenseur $\boldsymbol{V}$ est de taille $h \times l \times t$.

Rappelons ensuite que la transformée de Fourier $\mathcal{F}$ est un outil qui fournit une représentation fréquentielle d'un signal : cette dernière est strictement équivalente à celle,

temporelle, du signal original. Concrètement, soit $x(t)$ un signal unidimensionnel, alors sa transformée de Fourier peut se calculer via l'équation suivante :

$$\mathcal{F}(x) : f \longmapsto \int_{-\infty}^{+\infty} x(t) \exp\left(-2\pi i f t\right) \mathrm{d}t, \tag{7.1}$$

où i est le nombre imaginaire et f la fréquence.

Appliquons donc la transformée de Fourier $\mathcal{F}$ sur les données $\boldsymbol{V}$ selon la dimension temporelle, la troisième. Dans notre cas, cette étape permet de pouvoir observer des phénomènes répétitifs au niveau des pixels de l'image. La transformée de Fourier retournant un résultat complexe, nous prenons par la suite le module uniquement : ces valeurs informent de la présence et de la dominance d'une fréquence spécifique dans le signal d'origine. Nous obtenons ainsi la matrice réelle $\boldsymbol{X} = |\mathcal{F}(\boldsymbol{V})|$, toujours de taille $h \times l \times t$.

La dernière étape consiste à agréger l'information contenue dans le tenseur $\boldsymbol{X} = (x_{ijk})$. En l'occurence, nous construisons le vecteur colonne $\boldsymbol{y} = (y_1, \ldots, y_t)^T$ tel que

$$\forall k \in [\![1; t]\!] \quad y_k = \sum_{i=1}^{h} \sum_{j=1}^{l} x_{ijk}. \tag{7.2}$$

Ainsi, $\boldsymbol{y}$ résume le contenu fréquentiel de chaque pixel durant l'ensemble de la vidéo. Les sommations permettent d'accumuler les preuves sur certaines fréquences d'intérêt. De plus, le bruit s'étalant sur l'ensemble des fréquences, il se dissipe via ces sommes.

Appliquons cette méthode à une vidéo d'une remontée mécanique en fonctionnement normal : le résultat est donné en Figure 7.1. Notons d'abord que la fréquence nulle a été retirée du graphique par souçi de lisibilité : cette dernière représentant la somme des niveaux de gris moyens de chaque image, elle n'apporte aucune information pertinente. Soulignons également les fortes amplitudes, proches de 10^9, dûes à la somme sur l'ensemble des pixels de l'image. On distingue clairement sur cette figure un pic dominant ainsi que ses harmoniques. On peut extraire la fréquence d'intérêt via une simple recherche de maxima locaux et, en prenant les dix premiers pics pour plus de robustesse, on obtient une fréquence environ égale à 0,1404Hz. Pour convertir cette information en termes de nombre d'images, il suffit d'inverser cette fréquence et de la multiplier par le nombre d'images par seconde utilisé lors de l'acquisition de cette vidéo, égal à 26 ici. On obtient alors une durée d'un passage égal à 185,14 images.

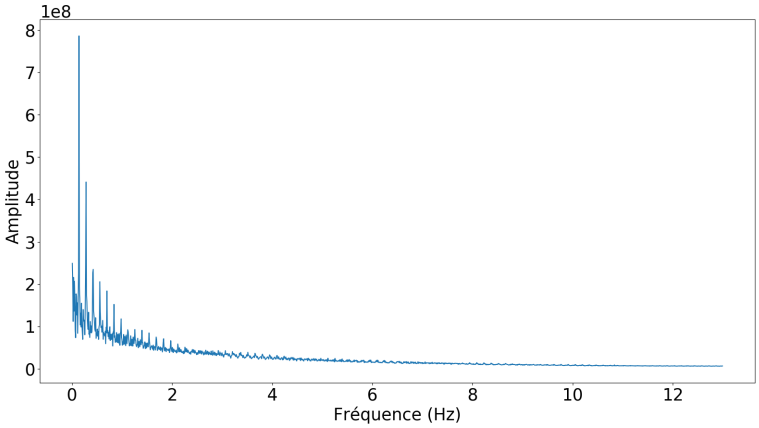


Figure 7.1. Représentation fréquentielle, après agrégation, d'une vidéo de remontée mécanique.

Quid de la réalité ? Grâce à une analyse visuelle de la vidéo concernée, nous avons trouvé une durée, moyennée sur les seize passages complets de la vidéo, environ égale à 184,69 images. Cette valeur est particulièrement proche de celle trouvée via la méthode proposée.

La technique détaillée ici fournit des résultats en accord avec toutes les remontées mécaniques sur lesquelles elle a été testée. Pour résumer, elle peut être une étape préliminaire, dès l'installation du système SIVAO sur un nouveau télésiège, afin de fournir des informations importantes concernant la remontée en question, notamment pour la modélisation automatique de la trajectoire des véhicules. Notons tout de même qu'à l'heure actuelle elle ne fonctionne que pour des régimes stationnaires, c'est-à-dire dans le cas où la vitesse de la remontée est constante. On peut toutefois imaginer appliquer cette solution sur des fenêtres glissantes dans le but de connaître la vitesse des véhicules à chaque instant : cela pourrait également servir d'indicateur *a posteriori* d'un danger, par exemple quand le télésiège subit un arrêt d'urgence.

7.3 Détection d'objets en mouvement via Déformation Optique

Nous proposons dans cette section un modèle de Flux Optique (FO) basé sur les contours pour la détection d'objets en mouvement. Les contours sont déterminés par la Déformation Optique (DO) qui correspond aux déformations du flux optique et qui est capable de détecter tous les mouvements d'une vidéo. Dans le but de distinguer le véritable mouvement du bruit (tel que les mouvements de feuilles d'arbres ou la

pluie/neige), nous proposons de travailler sur des fenêtres temporelles. Chacune de ces fenêtres est considérée comme l'instance minimale de notre modèle, ainsi nous calculons un FO unique grâce à l'Addition de Vecteurs (AV) qui permet de capturer les variations de mouvement longue distance, de la première à la dernière image de chaque fenêtre. L'idée sous-jacente est que le mouvement des objets ont un comportement stable à travers la fenêtre, alors que celui du bruit est erratique. De fait, les véritables mouvements sont accumulés, alors que le bruit est dissipé, permettant ainsi un seuillage simple afin de le supprimer.

Les principales contributions présentées dans cette section sont les suivantes :

- Détection des contours des objets en mouvement par utilisation de la DO.
- Inclusion de l'information temporelle grâce aux fenêtres glissantes.
- Utilisation de l'AV afin d'obtenir un FO unique sur chaque fenêtre.
- Seuillage adaptatif au niveau des fenêtres temporelles.

Le reste de cette section est organisé comme suit. Nous proposons en sous-section 7.3.1 un résumé des principales techniques d'extraction d'objets en mouvement dans une vidéo. En sous-section 7.3.2 est présenté le concept de la DO. Nous détaillons en sous-section 7.3.3 notre modèle et soulignons l'intérêt des fenêtres temporelles. Des expériences sont menées et des résultats sont donnés en sous-section 7.3.4. Nous concluons en sous-section 7.3.5 et présentons des perspectives sur ce travail.

7.3.1 Etat de l'art

La Détection d'Objets Mobiles (DOM) vise à récupérer le mouvement physique d'objets dans une scène vidéo. Elle reste un problème ouvert à cause du large éventail des scénarios réels pour lesquels elle peut être utilisée, tels que la vidéo surveillance, la recherche de trajectoires similaires, le suivi de traffic, parmi d'autres [Cos+16 ; Zha+12 ; HM09]. De plus, chacune de ces situations peut présenter différents challenges, *e.g.* variation d'illumination, occlusion partielle ou totale, arrière-plan complexe ou variation de dimension et d'apparence d'un objet.

Les algorithmes de DOM peuvent être classifiés suivant les différentes approches suivantes :

— Les modèles basés sur les différences temporelles appliquent une soustraction pixel à pixel entre images consécutives [Pau+17 ; AS+16 ; Rad+14]. Toutefois, ils échouent dans le cas de mouvement de forte amplitude ou de texture similaire entre le premier et l'arrière plan.

— Les méthodes de soustraction premier plan/arrière-plan visent à construire une modélisation de l'arrière-plan qui est ensuite soustrait à chaque image de la vidéo afin d'obtenir le premier plan [Cho+19 ; VR19 ; Hua+19 ; RG20 ; SG99]. Cependant, ces techniques sont coûteuses et requièrent un grand nombre d'images afin de déterminer un arrière-plan correct et robuste au bruit. De plus, des difficultés apparaissent dans le cas d'instabilité de la caméra.

— Les algorithmes de flux optique exploitent l'information de mouvement interne entre deux images consécutives [Tan+19 ; Kal+15 ; Tsa+16]. Elles ont récemment reçu un regain d'intérêt grâce à l'amélioration des modèles de flux optique [Gam+19 ; Pal+17 ; San+13].

A leur tour, les modèles de flux optique peuvent être séparés en ceux basés sur les régions, c'est-à-dire qu'ils visent à déterminer la forme des objets en mouvement [AMN13 ; PP14], et ceux basés sur les contours [YP05 ; PP12]. Notons toutefois que, en supposant que les contours soient bien détectés et complets, ces deux sous-catégories sont complémentaires, étant donné qu'il est aisé d'obtenir la forme complète d'un objet à partir de ses contours, et vice-versa. L'inconvénient majeur de ces méthodes est leur sensibilité au bruit.

7.3.2 Préliminaires

Il est connu qu'un fluide $\boldsymbol{u}$ donné peut être décomposé, dans un voisinage autour de chaque point, en la somme d'une translation (rigide), d'une déformation et d'une rotation (rigide) [CM90]. Par exemple, si l'on considère un point $\boldsymbol{x} \in \mathbb{R}^d$ et un voisin $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{h}$ avec $\boldsymbol{y}, \boldsymbol{h} \in \mathbb{R}^d$, on peut linéariser cette expression grâce au développement de Taylor :

$$\boldsymbol{u}(\boldsymbol{y}) = \boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x})\boldsymbol{h} + \mathrm{o}(|\boldsymbol{h}|), \tag{7.3}$$

avec $\boldsymbol{h} \to \mathbf{0}_d$, le vecteur nul en dimension d. $\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x})$ représente la matrice Jacobienne de la fonction $\boldsymbol{u}$ au point $\boldsymbol{x}$. Le premier terme représente la translation, alors que le second contient des informations sur la déformation et rotation du fluide. Comme toute matrice, $\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x})$ peut être décomposée en la somme d'une matrice symétrique et d'une matrice anti-symétrique. On peut donc écrire :

$$\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x}) = \underbrace{\frac{1}{2}\left(\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x})^T\right)}_{\varepsilon} + \underbrace{\frac{1}{2}\left(\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x}) - \boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{x})^T\right)}_{\eta}, \tag{7.4}$$

où ε est la partie symétrique, $\boldsymbol{\eta}$ celle anti-symétrique. Comme prouvé dans [CM90], ε inclut de l'information concernant la déformation, tandis que $\boldsymbol{\eta}$ décrit le mouvement rotationnel du fluide.

Dans notre cas, on peut considérer que le fluide $\boldsymbol{u}$ correspond au flux optique. Etant donné que nous nous intéressons à la déformation du mouvement, nous nous concentrons sur la partie symétrique de la matrice Jacobienne, également appelée DO dans la communauté de vision par ordinateur (*Optical Strain* en anglais).

Considérons le flux optique suivant :

$$\begin{array}{rccc} \boldsymbol{u} & : & \Omega \subset \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & & \begin{pmatrix} x \\ y \end{pmatrix} & \longmapsto & \begin{pmatrix} u_1(x,y) \\ u_2(x,y) \end{pmatrix} \end{array}, \tag{7.5}$$

où Ω représente l'espace des images. On peut alors obtenir une expression exacte de la DO comme suit :

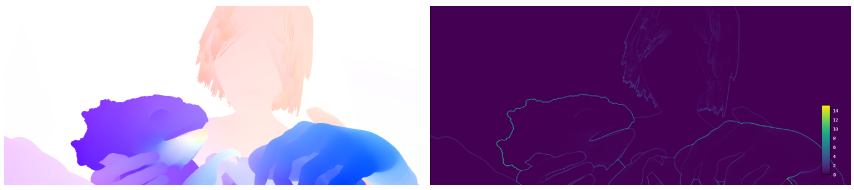
$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{yx} & \varepsilon_{yy} \end{pmatrix} = \begin{pmatrix} \frac{\partial u_1}{\partial x} & \frac{1}{2}\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) & \frac{\partial u_2}{\partial y} \end{pmatrix} \tag{7.6}$$

Par la suite, comme montré dans [Shr13], on peut calculer à partir de cette dernière équation la magnitude de la DO, à savoir

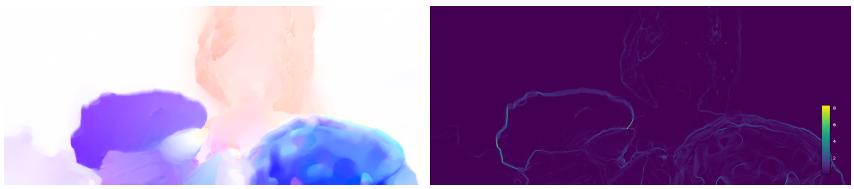
$$s = \sqrt{\varepsilon_{xx}^2 + 2\varepsilon_{xy}^2 + \varepsilon_{yy}^2}. \tag{7.7}$$

Ainsi, pour chaque pixel d'une image, il existe une matrice ε (de taille 2×2) et un scalaire s.

En Figure 7.2 sont présentés deux exemples d'un flux optique et de sa DO associée. Rappelons qu'en tout point de l'image, la direction du FO est représentée par la teinte, le module par la saturation (pour plus de précisions, voir [WC11]). Sur cette figure, on peut voir apparaître clairement les contours quand le flux optique est bien défini, comme c'est le cas pour la vérité-terrain. Précisons également que la vidéo utilisée dans ces exemples est synthétique, créée par ordinateur, ce qui permet de connaître la vérité-terrain sans besoin d'annotation manuelle. De plus, plus la magnitude du flux optique est grande, plus celle de la DO l'est également.



(a) Flux optique issu de la vérité-terrain.



(b) Flux optique obtenu grâce à l'algorithme TV-L1 [Zac+07 ; San+13]. Cette méthode fait office de référence dans le domaine.

Figure 7.2. Exemples de flux optique (à gauche) et DO associée (à droite). L'image est issue de la séquence BANDAGE_2 de la base de données SINTEL [But+12]. Le FO vérité-terrain est obtenu aisément étant donné que la scène est synthétique.

7.3.3 Méthode proposée

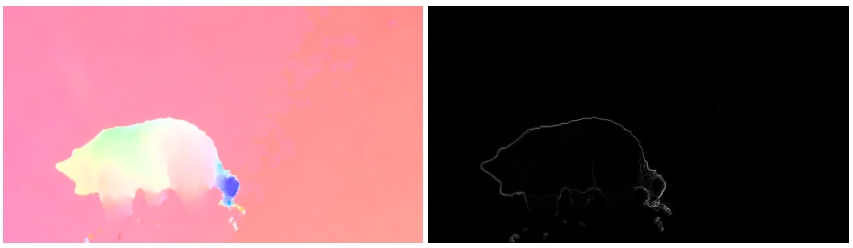
Nous détaillons ici un algorithme de détection d'objets en mouvement basé sur la DO et utilisant une approche par fenêtres temporelles.

Considérons une séquence d'images $I(\boldsymbol{x}, t) : \Omega \times \{1, \dots, T\} \longrightarrow \mathbb{R}^{\{1,3\}}$, où Ω est le domaine image et $\{1, \dots, T\}$ désigne l'ensemble discret des temps. Dénotons également $\boldsymbol{u}_t(\boldsymbol{x}) : \Omega \longrightarrow \mathbb{R}^2$ le flux optique entre les images $I(\boldsymbol{x}, t = k)$ et $I(\boldsymbol{x}, t = k + 1)$, c'est-à-dire le mouvement apparent entre un pixel $\boldsymbol{x} \in \Omega$ à l'instant k et celui correspondant à l'instant suivant $k + 1$.

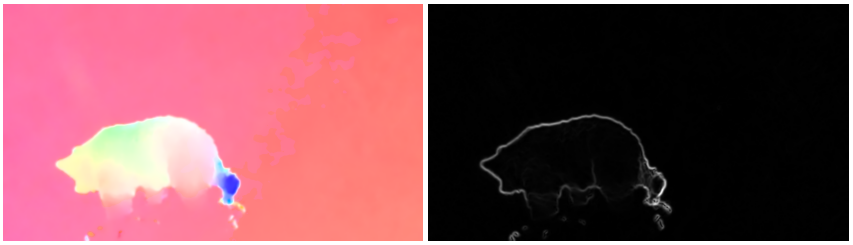
Afin de faciliter la détection des contours des objets en mouvement, nous proposons un filtrage du flux optique. En effet, comme on peut le voir en Figure 7.2, les frontières des objets peuvent être disjointes : ce problème peut être amoindri en appliquant un filtrage Gaussien sur chaque canal du FO, ce qui permet, par là même, de réduire le bruit. Nous choisissons, après expérimentations, un écart-type égal à 1,8. Les exemples donnés en Figure 7.3 montrent que les contours sont en effet plus marqués après ce filtrage.

Ensuite, nous proposons de travailler au niveau temporel au moyen de fenêtres glissantes d'images, et ce dans le but de pouvoir distinguer les objets en mouvement du bruit. Chacune de ces fenêtres, dénotées $\mathcal{W}_i$ dans la suite avec $i \in \mathbb{N}_*^+$, est constituée d'images de flux optique consécutives et est caractérisée par deux paramètres :

— le nombre n d'images incluses.



(a) Flux optique original.



(b) Filtrage Gaussien (écart-type = 1,8) appliqué au flux optique.

Figure 7.3. Effet du filtrage Gaussien sur le flux optique (à gauche) et la DO associée (à droite). L'image est issue de la séquence BEAR de la base de données DAVIS [Cae+19].

— le décalage σ entre deux fenêtres consécutives.

Un schéma de ce découpage est représenté en Figure 7.4. Ainsi, chaque fenêtre $\mathcal{W}_i$ contient les n images de flux optiques suivantes : $\{\boldsymbol{u}_{(i-1)\sigma+1}(\boldsymbol{x}), \ldots, \boldsymbol{u}_{(i-1)\sigma+n}(\boldsymbol{x})\}$. Notons attentivement que toutes les indexations (que ce soit pour les images ou pour les fenêtres) débutent à 1.

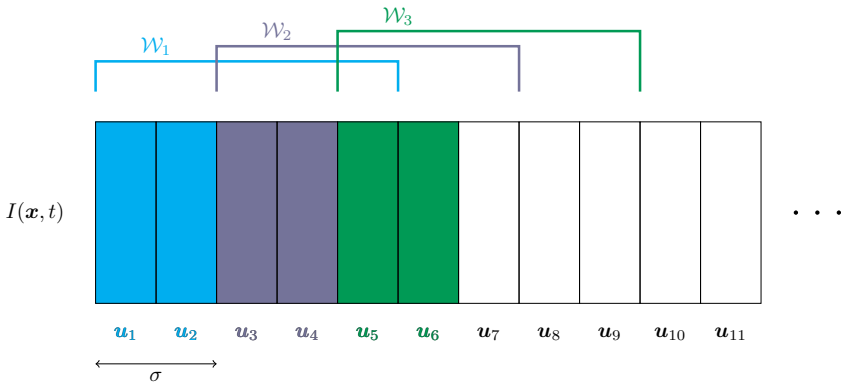


Figure 7.4. Schéma représentant le découpage en fenêtres d'une vidéo, utilisé pour l'algorithme proposé. Ici, $n = 5$ et $\sigma = 2$.

L'organisation en fenêtres temporelles autorise un travail sur des sous-séquences durant lesquelles le véritable mouvement est accumulé, engendrant des valeurs élevées pour la magnitude de la DO, à l'opposé du bruit qui aboutit à des valeurs faibles. De

cette manière, il est possible de séparer le mouvement des objets présents dans la scène du bruit via un processus en trois étapes. D'abord, un seuil est déterminé en prenant en compte les différents mouvements de chaque fenêtre $\mathcal{W}_i$. Ensuite, le seuil est appliqué individuellement à chaque image. Enfin, une opération de morphologie mathématique est utilisée afin de supprimer le bruit résiduel. Ces trois étapes sont détaillées dans la suite de cette sous-section.

Calcul du seuil

Le flux optique informe du mouvement entre deux images consécutives : ce dernier peut alors être bruité, à cause de la scène ou de l'algorithme lui-même. Dans l'optique d'obtenir un seuil unique pour chaque fenêtre $\mathcal{W}_i$, il serait désirable d'avoir un flux optique de la première image vers chacune des autres images de $\mathcal{W}_i$. De cette manière, on obtiendrait des valeurs élevées pour les objets ayant un mouvement régulier, des valeurs faibles dans le cas contraire, c'est-à-dire en présence de bruit. Cependant, il est connu que les méthodes de flux optique échouent sur des larges distances, comme on peut le voir en Figure 7.5b.

(a) Vérité-terrain. **(b)** Flux optique TV-L1 entre l'image de départ et celle d'arrivée. **(c)** AV sur le flux optique TV-L1 entre les deux images.

Figure 7.5. Comparaison entre le flux optique sur de grandes distances et l'AV. Les images (n°1 et n°8) sont issues de la séquence ALLEY_1 de la base de données SINTEL [But+12].

En revanche, pour bénéficier de l'accumulation temporelle du mouvement, nous déterminons les vecteurs de mouvement entre deux images non-consécutives grâce à l'idée proposée dans [Shr13]. Baptisée Addition de Vecteurs (AV), elle consiste à calculer un flux optique unique, à partir de la première jusqu'à la dernière image de chaque fenêtre, grâce à l'équation récursive suivante :

$$\begin{cases} \boldsymbol{v}_k(\boldsymbol{x}) = \boldsymbol{u}_k(\boldsymbol{x}) & \text{si } k = 1 \\ \boldsymbol{v}_k(\boldsymbol{x}) = \boldsymbol{v}_{k-1}(\boldsymbol{x}) + \boldsymbol{u}_k\left(\boldsymbol{x} + \boldsymbol{v}_{k-1}(\boldsymbol{x})\right) & \text{si } k \geq 2 \end{cases} \tag{7.8}$$

où $\boldsymbol{v}$ est l'AV et k correspond à l'indice de l'image. Une illustration de l'AV (Équation 7.8) est donnée en Figure 7.6. On considère trois images consécutives (F_t, F_{t+1} et F_{t+2}) et un point spécifique (croix rouge) d'un cercle jaune en mouvement. Le

vecteur de mouvement entre F_t et F_{t+2} est déterminé en additionnant $\boldsymbol{u}(F_t, F_{t+1})$ et $\boldsymbol{u}(F_{t+1}, F_{t+2})$, d'où le nom de la technique.

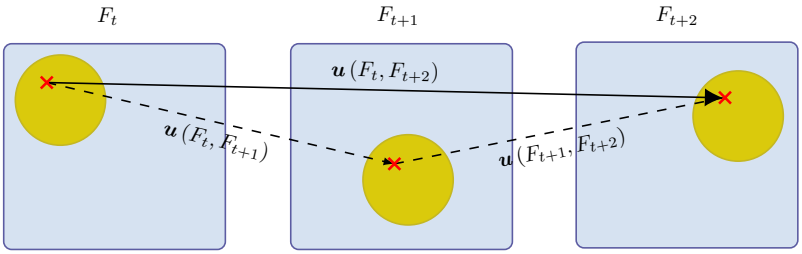


Figure 7.6. Illustration de l'AV sur un exemple simple. Les mouvements entre la première et la deuxième image, ainsi qu'entre la deuxième et la troisième permettent de directement déduire celui entre la première et la troisième.

Comme on peut l'observer en Figure 7.5, l'AV montre de meilleures performances que le calcul du flux optique entre deux images non-consécutives (ici la première image de la séquence et la huitième). Contrairement au FO, l'addition de vecteurs prend en compte l'information de mouvement entre chaque paire d'images.

Par la suite, la magnitude de la DO est calculée sur chaque AV de la fenêtre, grâce à l'Équation 7.6. Concrètement, nous obtenons une succession de matrices représentant les magnitudes pour chaque fenêtre $\mathcal{W}_i$: $\{\boldsymbol{s}_{(i-1)\sigma+1}(\boldsymbol{x}), \ldots, \boldsymbol{s}_{(i-1)\sigma+n}(\boldsymbol{x})\}$. Rappelons qu'une valeur de magnitude est disponible pour chaque pixel, ainsi nous avons bel et bien affaire ici à des matrices, de même taille que les images de la vidéo. De plus, la fenêtre $\mathcal{W}_i$ contient autant d'images de flux optique que de DO.

Une fois les magnitudes pour l'ensemble de la fenêtre calculées, il est nécessaire de déterminer le seuil τ_i associé à la fenêtre $\mathcal{W}_i$ qui permet de supprimer les faibles magnitudes et donc de ne garder que les informations pertinentes à propos des contours des objets. En suivant l'idée de [Lio+16], nous proposons le seuil adaptatif suivant :

$$\tau_i = \min_j \min_{\boldsymbol{x}\in\Omega} s_j(\boldsymbol{x}) + \rho \left(\max_j \max_{\boldsymbol{x}\in\Omega} s_j(\boldsymbol{x}) - \min_j \min_{\boldsymbol{x}\in\Omega} s_j(\boldsymbol{x}) \right), \tag{7.9}$$

avec $\rho \in]0;1[$. Ce paramètre est un percentile à l'intérieur de la plage des magnitudes. Par exemple, choisir $\rho = 0,05$ équivaut à enlever 5% des plus faibles valeurs, correspondant à des mouvements liés au bruit.

Le processus complet visant au calcul du seuil de chaque fenêtre est résumé en Algorithme 4.

Input: Séquences d'images $I_1, I_2, \ldots$

Déterminer le FO $\boldsymbol{u}$ entre chaque paire d'images consécutives
Appliquer un filtre Gaussien sur chaque canal du FO
Séparer le FO en fenêtres $\mathcal{W}_i$ selon les paramètres n et σ ▷ Figure 7.4
for *chaque fenêtre* $\mathcal{W}_i$ **do**
    for *chaque image de FO* $\boldsymbol{u_j}$ *dans* $\mathcal{W}_i$ **do**
        Calculer l'AV entre $\boldsymbol{u_1}$ et $\boldsymbol{u_j}$ ▷ Équation 7.8
        Déduire la magnitude $\boldsymbol{s_j}$ de la DO ▷ Équation 7.7
    end
    Obtenir le seuil de filtrage τ_i ▷ Équation 7.9
end

Algorithme 4: Acquisition du seuil pour chaque fenêtre.

Application du seuil

Nous appliquons ici les valeurs de seuil obtenues précédemment pour chaque fenêtre à chaque image. Etant donné que $\sigma < n$, chaque image de DO appartient à plusieurs fenêtres et donc plusieurs seuils lui sont associés.

Nous proposons d'obtenir un unique seuil pour chaque images DO de la vidéo en utilisant une **moyenne mobile pondérée** appliquée aux seuils des différentes fenêtres. En particulier, pour une image spécifique, le poids associé à chaque seuil dépend du degré d'appartenance de ladite image à la fenêtre correspondante. Cette information est fournie par la position de l'image par rapport à la fin de chaque fenêtre à laquelle elle appartient. Ainsi, un plus grand poids est donné aux premières images de chaque fenêtre, un plus petit aux dernières. Ce choix est motivé par l'emploi de l'AV (voir Équation 7.8) qui, pour rappel, construit un flux optique unique en prenant la première image de chaque fenêtre comme référence. En utilisant ce type de moyennage, on garantit que plus d'importance est donnée aux premières images, en lesquelles on a le plus confiance.

Concrètement, pour chaque image t de la vidéo, le seuil associé $\tilde{\tau}_t$ peut être déterminé comme suit :

$$\tilde{\tau}_t = \frac{\sum_{i=a}^{b}(1+(i-1)\sigma+n-t)\tau_i}{\sum_{i=a}^{b}(1+(i-1)\sigma+n-t)}, \tag{7.10}$$

où n représente le nombre d'images dans chaque fenêtre, σ est celui entre deux fenêtres consécutives, τ_i correspond au seuil de la $i^{\text{ème}}$ fenêtre, a est l'indice de la première fenêtre à laquelle l'image t appartient et b l'indice de la dernière. Plus précisément, on peut déterminer ces deux dernières valeurs par :

$$a = \max\left(\left\lceil \frac{t-n}{\sigma} \right\rceil, 0\right) + 1, \tag{7.11}$$

$$b = \left\lfloor \frac{\min(t, T) - 1}{\sigma} \right\rfloor + 1, \tag{7.12}$$

avec T le nombre total d'images présentes dans la vidéo et $\lceil . \rceil$ (respectivement $\lfloor . \rfloor$) la partie fractionnaire (respectivement entière) d'un nombre réel. A titre d'illustration, considérons à nouveau le schéma de la Figure 7.4, pour lequel $n = 5$ et $\sigma = 2$, et plus particulièrement l'image $\boldsymbol{u}_5$. Dans ce cas, $a = \max\left(\left\lceil \frac{5-5}{2} \right\rceil, 0\right) + 1 = 1$ et $b = \left\lfloor \frac{\min(5,100)-1}{2} \right\rfloor + 1 = 3$: effectivement, on confirme visuellement que cette image appartient bien aux fenêtres $\mathcal{W}_1$, $\mathcal{W}_2$ et $\mathcal{W}_3$. En conséquence, le seuil filtré peut s'exprimer ainsi :

$$\tilde{\tau}_5 = \frac{1 \times \tau_1 + 3 \times \tau_2 + 5 \times \tau_3}{9}. \tag{7.13}$$

Comme dit ci-dessus, il est clair que le seuil de la fenêtre $\mathcal{W}_3$ pèse plus étant donné que $\boldsymbol{u}_5$ est la première image de cette même fenêtre.

Morphologie mathématique

L'image résultante, après seuillage, peut encore contenir de petites zones bruitées. Afin de les supprimer définitivement, nous proposons d'utiliser un opérateur de morphologie mathématique, c'est-à-dire une érosion du premier-plan suivi par une dilatation. L'élément structurant choisi empiriquement ici est un disque de rayon trois pixels.

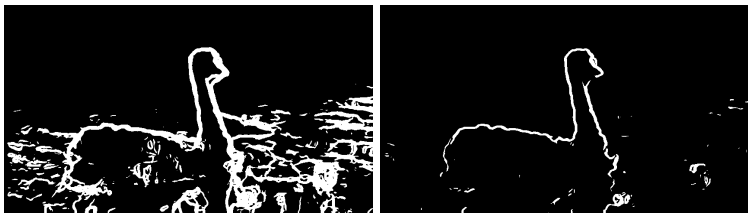
7.3.4 Résultats expérimentaux

Dans cette sous-section sont présentées les expériences menées et les résultats obtenus par notre méthode ainsi que deux autres modèles dans l'optique de la détection d'objets en mouvement.

Intérêt du découpage temporel

Commencons d'abord par souligner le bénéfice apporté par les fenêtres glissantes. On donne en Figure 7.7 deux images représentant la magnitude de la DO calculée pour des tailles de fenêtre différentes : $n = 1$ à gauche, $n = 10$ à droite. Si la taille de la fenêtre est égale à 1, cela signifie que les seuils sont déterminés et appliqués image par image. On peut voir sur la figure de gauche que le bruit reste, notamment les ondelettes sur l'eau. Comme expliqué précédemment, sans

l'information temporelle, tous les mouvements ont la même importance, même le bruit (c'est-à-dire les mouvements ponctuels).



(a) $n = 1$ et $\sigma = 1$. **(b)** $n = 10$ et $\sigma = 5$.

Figure 7.7. Influence de la taille de la fenêtre sur la DO. L'image est issue de la séquence BLACKSWAN de la base de données DAVIS [Cae+19].

Modèles comparés

Notre algorithme est comparé à deux autres méthodes utilisant elles aussi le flux optique en entrée afin de détecter les objets en mouvement dans une scène. Toutes deux opèrent sur le domaine spatial en ne prenant qu'une seule image en compte à la fois.

— La première méthode, décrite par Patel et Parmar [PP14] en 2014, détermine les contours directement grâce au flux optique. Ils proposent ensuite une technique de seuillage basée sur la moyenne et l'écart-type de chaque canal du FO. Enfin, des opérations de morphologie mathématique sont appliquées. Dans la suite, nous désignerons cet algorithme par Statistical Optical Flow Thresholding (SOFT).

— Le deuxième modèle, proposé par Tang *et al.* [Tan+19] et nommé Entropy-based Canny operator with Local and Global Optical Flow (EC-LGOF), fonctionne comme suit. Les contours des objets sont d'abord déterminés directement sur les images originales et le seuillage proposé dans [KI86] y est appliqué. Ensuite, la méthode de seuillage Otsu [Ots79] est exécutée sur la magnitude du flux optique. Par suite, les contours apparaissant dans les deux images (originale et flux optique) sont considérés. Enfin, l'image finale est dilatée.

Bases de données utilisées

Les modèles de détection d'objets en mouvement sont évalués sur trois bases de données réelles :

- **DAVIS** [Cae+19]
  Elle contient 90 séquences, allant de 25 à 104 images (médiane = 71), acquises avec une caméra non-statique, comprenant un à quatre objets d'intérêt. Les difficultés principales liées à ces données sont la non-rigidité des objets, les contours flous à cause du mouvement, les occlusions partielles ou totales, les objets sortant du cadre, les vibrations de la caméra, les objets hétérogènes (en termes de texture), le dynamisme de l'arrière-plan, *etc*.

- **SBI 2015** [MP15]
  Elle consiste en treize séquences de 90 à 740 images (médiane = 350). Les deux challenges majeurs de cette base de données sont la faible résolution des images et la présence d'objets encombrants, occupant la plupart du domaine spatial.

- **Remontées mécaniques**
  Elle est constituée de 76 séquences de 40 images chacune pour plusieurs remontées mécaniques de différentes stations. Pour rappel, la complexité de cette base de données repose sur des conditions météorologiques variées (soleil, blizzard, nuages), des saisons différentes (été et hiver principalement), des occlusions partielles ou totales, des objets sortant du cadre, des objets dont la taille varie, des objets de formes complexes (*i.e.* avec des contours fins et des trous), des variations d'illumination, des vibrations de caméra, *etc*. De plus, la vérité-terrain associée à cette base de données a été déterminée manuellement grâce à une annotation pixel à pixel : des détails sont donnés en Appendice D.

Méthodologie de comparaison

Comme expliqué en sous-section 7.3.3, notre algorithme retourne la frontière des objets en mouvement dans la scène. Cependant, la vérité-terrain fournie avec les bases de données détaillées précédemment offre la forme complète de chaque objet, pixel à pixel. Ainsi, afin de présenter une comparaison équitable entre les différents algorithmes, nous proposons de déterminer la boîte englobante de toutes les sorties d'algorithmes, également pour les objets de la vérité-terrain.

Chaque boîte englobante est calculée comme suit. Supposons que l'image de sortie retournée par une des techniques ne contienne qu'une composante connexe, *i.e.* qu'un seul objet, composée de m pixels. Cette dernière peut alors être définie

comme : $\mathcal{C} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. La boîte englobante associée est ainsi égale au rectangle dont les coins sont :

$$x_{\min} = \min_i x_i, \tag{7.14}$$

$$x_{\max} = \max_i x_i, \tag{7.15}$$

$$y_{\min} = \min_i y_i, \tag{7.16}$$

$$y_{\max} = \max_i y_i. \tag{7.17}$$

Si l'image comprend plus d'une composante connexe, alors le même processus est appliqué pour chacune, c'est-à-dire qu'une boîte englobante est déterminée pour chaque composante. En particulier, pour les deux modèles comparés et notre algorithme, le nombre d'objets est donné en entrée, en supposant que ce dernier est égal au nombre de composantes connexes trouvées dans l'image. La Figure 7.8 donne quelques exemples pratiques concernant les boîtes englobantes, avec un ou plusieurs objets, sur la vérité-terrain ou sur les images de DO obtenues en sortie de notre méthode.

(a) Images issues de la vérité-terrain.

(b) Images retournées par l'algorithme proposé.

Figure 7.8. Illustration de la détermination des boîtes englobantes (rectangles roses). Les images sont issues des séquences BUS et CROSSING de la base de données DAVIS [Cae+19].

Dans la suite, la comparaison entre boîtes englobantes se fait via les Prédictions positives correctes (P), le Rappel (R), la F-mesure [Chi92] (F) et l'indice de Jaccard

(J). P est égal à la proportion de pixels positifs correctement détectés, R à celle de pixels (positifs et négatifs) correctement étiquetés. Ainsi :

$$P = \frac{VP}{VP + FP}, \tag{7.18}$$

$$R = \frac{VP}{VP + FN}, \tag{7.19}$$

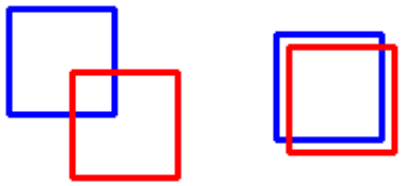
où VP dénote le nombre de pixels considérés comme faisant partis du premier-plan dans la vérité-terrain ainsi que dans la prédiction, FP le nombre de pixels prédits comme premier-plan mais qui appartiennent à l'arrière-plan en réalité, FN le nombre de pixels prédits comme arrière-plan alors qu'ils appartiennent au premier-plan. Le F-score est défini comme la moyenne harmonique de P et de R :

$$F = 2 \cdot \frac{P \times R}{P + R}. \tag{7.20}$$

Quant à l'indice de Jaccard, il représente à quel point la boîte englobante prédite s'aligne par rapport à celle de la vérité-terrain. Plus précisément :

$$J = \frac{VP}{VP + FN + FP}. \tag{7.21}$$

Une interprétation géométrique de cette mesure est donnée en Figure 7.9.



(a) $J \approx 0,637.$ **(b)** $J \approx 0,093.$

Figure 7.9. Exemples de boîtes englobantes et indice de Jaccard correspondant.

Enfin, pour pouvoir comparer équitablement les trois méthodes, nous utilisons le même flux optique en entrée, à savoir le modèle TV-L1 [San+13] grâce à sa rapidité et ses bonnes performances. En ce qui concerne les algorithmes EC-LGOF et SOFT, nous employons les paramètres proposés dans les articles correspondants. Quant au nôtre, pour chaque séquence, différentes tailles de fenêtre sont testées (entre $n = 2$ et $n = 20$) : celle engendrant la F-mesure la plus élevée est gardée en mémoire. Nous faisons référence ici à la mesure moyennée sur l'ensemble des images de la séquence. De plus, le décalage est fixé à la moitié de la taille de fenêtre, c'est-à-dire $\sigma = \lfloor n/2 \rfloor$.

Résultats et analyse

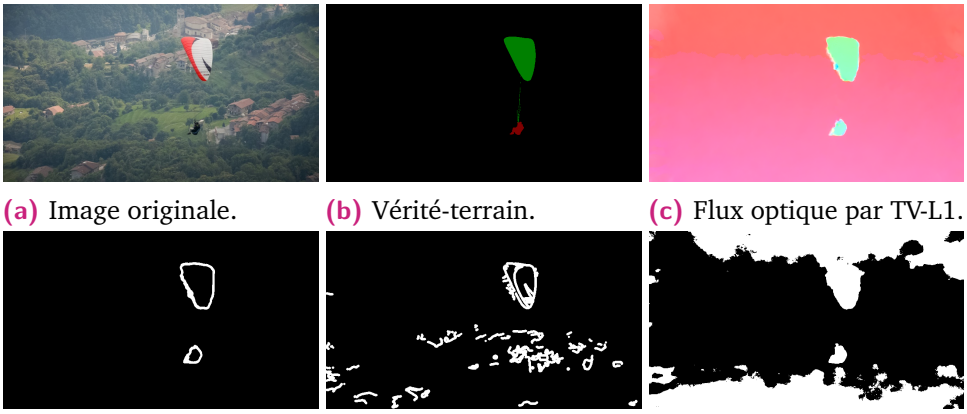
Les Tableaux 7.1, 7.2 et 7.3 montrent un extrait des résultats, PPC (P), Rappel (R), F-score (F) et indice de Jaccard (J), obtenus par notre méthode ainsi que deux autres algorithmes sur les bases de données DAVIS, SBI 2015 et remontées mécaniques. La dernière ligne de chaque tableau correspond à la métrique moyenne sur la totalité des séquences de chaque jeu de données. Les meilleures performances sont affichées en gras.

On peut observer que notre algorithme surclasse les autres sur toutes les bases de données en termes de F-mesure et d'indice de Jaccard. Cela peut s'expliquer par le fait que notre modèle se concentre sur la suppression de bruit. Egalement, de bonnes performances sont obtenues en matière de PPC, car la technique proposée est capable de détecter correctement les contours des objets en mouvement. En ce qui concerne le Rappel, il représente le taux de VP, autrement dit il mesure la capacité d'un algorithme à détecter les pixels appartenant au premier-plan. Un Rappel de 1 est facilement atteignable en considérant que l'ensemble des pixels constituent le premier-plan, même si cette solution est à proscrire étant donné qu'elle ne représente pas réellement la scène. C'est ce qui se passe avec la méthode SOFT : en effet, on peut voir en Figure 7.10f qu'elle considère la plupart des pixels de l'image comme faisant partie du premier-plan, donnant une valeur de Rappel élevée, dans ce cas parfaite dûe au fait que le calcul du score est effectué sur les boîtes englobantes.

En ce qui concerne la taille de fenêtre choisie par validation croisée, on peut observer qu'elle reste plutôt constante pour les bases de données SBI 2015 et remontées mécaniques, contrairement à DAVIS. Ceci est dû au fait que les deux premiers jeux de données contiennent des objets se déplacant à vitesse quasi-constante, ce qui n'est pas le cas pour DAVIS. De plus, nous avons pu observer que des scènes avec des objets dont la vitesse relative est grande abouttisaient à des valeurs plus élevées pour n.

Nous présentons cinq exemples visuels en Figures 7.10, 7.11, 7.12, 7.13 et 7.14. Dans le premier cas (Figure 7.10), notre méthode réussit à capturer seulement l'objet en mouvement et à complètement supprimer le bruit, à l'opposé des autres algorithmes. L'élimination des mouvements liés au bruit est également visible en Figure 7.12 et Figure 7.13. Un comportement similaire est observable en Figure 7.11 : la technique proposée donne de bons résultats comparés aux autres dans les cas de caméras non-statiques ou de mouvements générés par des objets en arrière-plan. Enfin, la Figure 7.14 montre à quel point la première étape de notre méthode est cruciale. En effet, l'obtention d'un flux optique correct est un élément clé : les contours des

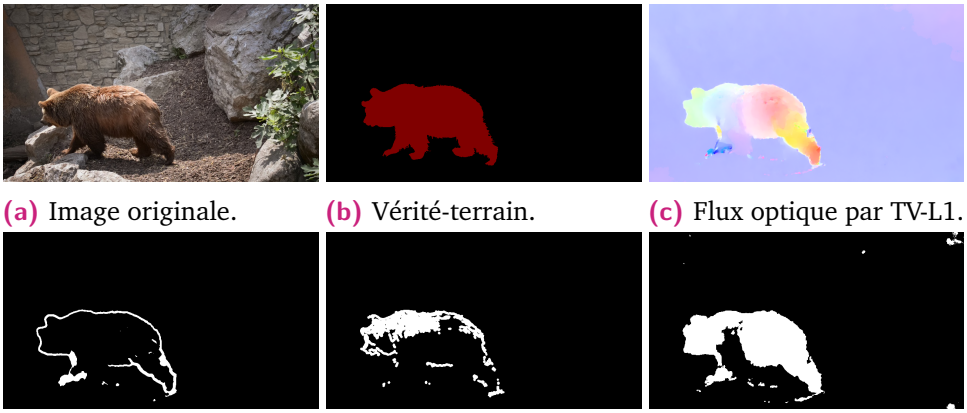
différents objets ne sont pas récupérés proprement si le flux optique n'arrive pas à distinguer les objets du premier-plan et de l'arrière-plan.



(a) Image originale. **(b)** Vérité-terrain. **(c)** Flux optique par TV-L1.

(d) Résultat par DO. **(e)** Résultat par EC-LGOF. **(f)** Résultat par SOFT.

Figure 7.10. Image originale, vérité-terrain, flux optique et résultats des trois méthodes. L'image est issue de la séquence PARAGLIDING de la base de données DAVIS.



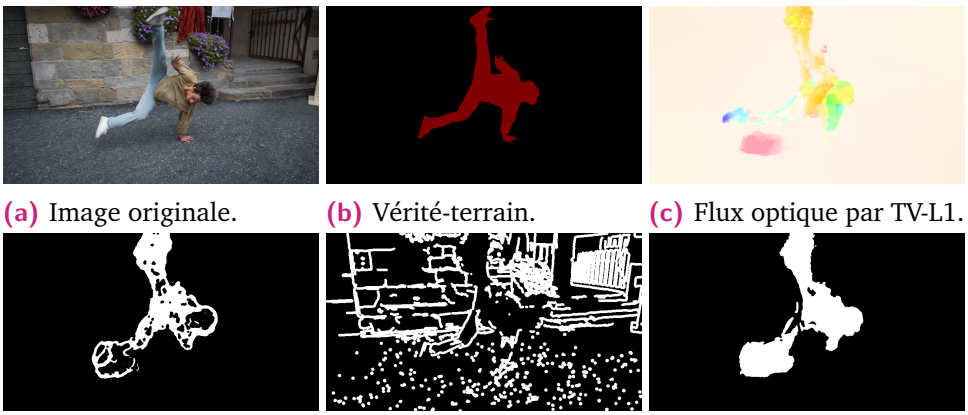
(a) Image originale. **(b)** Vérité-terrain. **(c)** Flux optique par TV-L1.

(d) Résultat par DO. **(e)** Résultat par EC-LGOF. **(f)** Résultat par SOFT.

Figure 7.11. Image originale, vérité-terrain, flux optique et résultats des trois méthodes. L'image est issue de la séquence BEAR de la base de données DAVIS.

7.3.5 Conclusion et perspectives

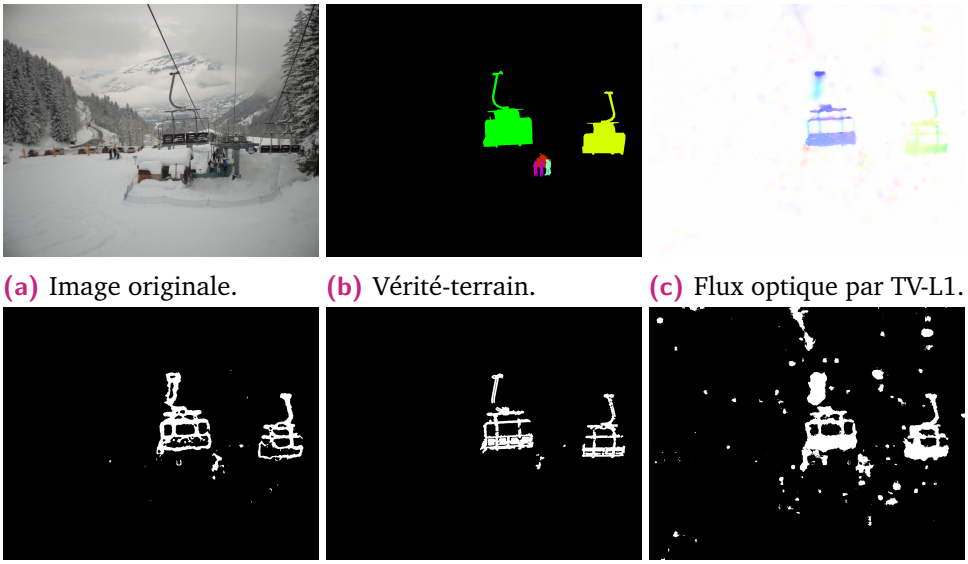
Nous avons proposé dans cette section un modèle permettant de détecter les contours d'objets en mouvement dans les scènes non-contraintes. Pour résumer sommairement, il consiste en un filtrage du flux optique. Concrètement, la méthode fonctionne en déterminant la DO, qui correspond à la déformation du flux optique. Les contours sont ensuite extraits à partir de l'information apportée par la DO. De plus, nous avons proposé d'inclure l'évolution temporelle de ces contours via l'ajout de fenêtres glissantes, permettant ainsi un seuillage local temporellement. Notre méthode



(a) Image originale. (b) Vérité-terrain. (c) Flux optique par TV-L1.

(d) Résultat par DO. (e) Résultat par EC-LGOF. (f) Résultat par SOFT.

Figure 7.12. Image originale, vérité-terrain, flux optique et résultats des trois méthodes. L'image est issue de la séquence BREAKDANCE de la base de données DAVIS.



(a) Image originale. (b) Vérité-terrain. (c) Flux optique par TV-L1.

(d) Résultat par DO. (e) Résultat par EC-LGOF. (f) Résultat par SOFT.

Figure 7.13. Image originale, vérité-terrain, flux optique et résultats des trois méthodes. L'image est issue d'une séquence de la base de données MIVAO.

montre des résultats encourageants lorsque qu'elle est comparée à des algorithmes similaires. Il est en effet important de noter que notre approche surclasse en termes de PPC et F-score. Le travail présenté ici a donné lieu à une publication :

> M. Oliver-Parera, J. Muzeau, P. Ladret et P. Bertolino, "Contour Detection of Multiple Moving Objects in Unconstrained Scenes using Optical Strain", International Conference on Digital Image Computing : Techniques and Applications, 2020 (en attente de publication).

(a) Image originale. **(b)** Vérité-terrain. **(c)** Flux optique par TV-L1.

(d) Résultat par DO. **(e)** Résultat par EC-LGOF. **(f)** Résultat par SOFT.

Figure 7.14. Image originale, vérité-terrain, flux optique et résultats des trois méthodes. L'image est issue de la séquence PEOPLE & FOLIAGE de la base de données SBI 2015.

Toutefois, plusieurs pistes d'amélioration peuvent être considérées. La première consiste à étendre le modèle détaillé dans cette section afin d'inclure l'information spatiale. Par exemple, on pourrait utiliser des statistiques spécifiques afin de décider d'un seuil adaptatif pour différentes zones de l'image. Une autre possibilité est la création d'heuristiques dans le but d'aider au choix des deux paramètres de notre modèle, à savoir la taille et le décalage entre deux fenêtres consécutives. Nous avons en effet déjà pu observer que cette sélection dépend fortement des magnitudes du flux optique de la scène. De plus, la méthode de seuillage utilisée ici étant simpliste, un autre choix pourrait donner lieu à de meilleurs résultats. Enfin, il pourrait être intéressant d'intégrer une technique de suivi de contours à travers le temps, ce qui permettrait d'accroître la robustesse de l'algorithme proposé.

7.4 Conclusion

Ce chapitre s'est consacré à l'extraction des images unitaires à partir de celles issues de la caméra, plus précisément à la détection des véhicules et à leur suivi au fil du temps. Rappelons que les travaux présentés ici n'utilisent pas d'information *a priori*, telle qu'un modèle du joint de chaque type de véhicule créé manuellement.

Nous avons d'abord proposé une méthode, basée sur l'étude fréquentielle d'une vidéo, permettant de déterminer automatiquement la durée d'un passage. On peut considérer appliquer cette dernière dès l'installation d'une caméra sur une nouvelle rémontée mécanique afin de recueillir des informations sur la physique du télésiège nécessaires au bon fonctionnement du système.

Ensuite, un algorithme de détection d'objets en mouvement a été présenté. Il consiste, pour résumer, à filtrer le flux optique en étudiant sa déformation temporelle, également appelée DO. La proposition montre des résultats satisfaisants et de nombreuses pistes d'amélioration existent.

Une autre idée envisagée est l'utilisation de régulateur Proportionnel-Intégral-Dérivé (PID), un pour chaque pixel de l'image, afin d'obtenir une estimation de l'arrière-plan à chaque instant. Elle pourrait notamment être intéressante pour contrôler les vibrations de la caméra. Les premières expérimentations dans cette direction montrent des résultats prometteurs.

La perspective principale liée à ces travaux est la modélisation de la trajectoire de n'importe quelle remontée mécanique. Concrètement, nous envisageons de concaténer les images consécutives du premier-plan, obtenues via l'algorithme détaillé en section 7.3. Ceci donnerait lieu à une représentation en trois dimensions (deux dimensions pour l'image et une pour le temps). Ainsi, les évolutions temporelles de chaque véhicule et de sa projection perspective sont prises en compte. On pourrait par la suite imaginer obtenir automatiquement le masque du joint à chaque instant pour assurer une meilleure détection et suivi. Egalement, cela nous permettrait de pouvoir localiser le véhicule à n'importe quel instant.

Séquence	DO				EC-LGOF				SOFT			
	P	R	F	J	P	R	F	J	P	R	F	J
BEAR ($n = 5$)	**0,801**	0,960	**0,868**	**0,776**	0,563	0,993	0,667	0,558	0,232	**1**	0,364	0,232
BLACKSWAN ($n = 20$)	**0,477**	0,968	0,633	**0,468**	0,450	**1**	0,614	0,450	0,305	**1**	0,467	0,305
BMX-BUMPS ($n = 18$)	**0,320**	0,780	**0,397**	**0,297**	0,277	0,732	0,335	0,252	0,209	**0,838**	0,294	0,208
BREAKDANCE ($n = 20$)	**0,403**	0,995	**0,563**	**0,402**	0,229	**1**	0,369	0,229	0,345	0,998	0,503	0,344
BUS ($n = 20$)	**0,876**	0,972	**0,914**	**0,848**	0,810	0,992	0,883	0,805	0,371	**1**	0,537	0,371
CAMEL ($n = 2$)	**0,935**	0,951	**0,938**	**0,890**	0,746	0,971	0,779	0,719	0,307	**0,999**	0,454	0,307
DISC-JOCKEY ($n = 2$)	**0,886**	0,440	0,539	0,419	0,788	0,278	0,402	0,261	0,855	**0,716**	**0,751**	**0,615**
DRIFT-CHICANE ($n = 20$)	**0,135**	0,967	**0,218**	**0,134**	0,025	**1**	0,048	0,025	0,028	**1**	0,053	0,028
DRIFT-TURN ($n = 19$)	0,392	0,976	0,511	0,377	**0,458**	0,932	**0,570**	**0,426**	0,259	**0,998**	0,357	0,258
ELEPHANT ($n = 15$)	**0,942**	0,957	**0,948**	**0,902**	0,596	0,987	0,682	0,585	0,234	**0,998**	0,372	0,234
KOALA ($n = 4$)	**0,826**	0,892	**0,830**	**0,748**	0,646	0,933	0,723	0,582	0,722	**0,967**	0,799	0,693
LAB-COAT ($n = 2$)	**0,557**	0,662	**0,557**	0,413	0,419	0,738	0,443	0,297	0,448	**0,954**	0,552	**0,444**
LUCIA ($n = 8$)	**0,824**	0,917	**0,857**	**0,757**	0,799	0,826	0,751	0,645	0,498	**0,966**	0,624	0,483
Moyenne	**0,638**	0,804	**0,645**	**0,521**	0,443	0,828	0,480	0,360	0,401	**0,900**	0,477	0,355

Table 7.1. Scores (P, R, F et J) obtenus par trois méthodes dont la nôtre, dénotée DO, sur treize séquences de la base de données DAVIS. La dernière ligne donne la moyenne sur l'ensemble des 90 séquences.

Séquence	DO				EC-LGOF				SOFT			
	P	R	F	J	P	R	F	J	P	R	F	J
Board ($n = 2$)	0,869	**0,859**	**0,839**	**0,743**	**0,915**	0,648	0,721	0,607	0,900	0,775	0,814	0,706
CAVIAR 1 ($n = 2$)	**0,748**	0,259	0,352	0,230	0,742	0,246	0,326	0,205	0,535	**0,587**	**0,490**	**0,344**
CAVIAR 2 ($n = 3$)	0,610	0,479	**0,425**	**0,308**	**0,640**	0,464	0,406	0,297	0,091	**0,698**	0,141	0,079
Ca Vignal ($n = 2$)	0,431	0,707	0,492	0,378	**0,530**	0,822	**0,582**	**0,483**	0,255	**0,925**	0,380	0,249
Candela m1.10 ($n = 2$)	0,520	0,355	0,356	0,276	**0,638**	0,445	0,447	0,330	0,442	**0,754**	**0,457**	**0,334**
Foliage ($n = 2$)	0,956	**0,975**	**0,958**	**0,933**	**0,967**	0,838	0,873	0,812	0,960	0,966	0,955	0,929
Hall & Monitor ($n = 2$)	0,806	0,562	**0,605**	**0,455**	**0,817**	0,536	0,562	0,428	0,259	**0,849**	0,367	0,233
Highway I ($n = 2$)	0,811	**0,794**	**0,773**	**0,653**	**0,820**	0,579	0,636	0,492	0,799	0,773	0,744	0,624
Highway II ($n = 2$)	0,877	0,316	0,429	0,292	**0,904**	0,236	0,323	0,213	0,821	**0,435**	**0,478**	**0,337**
Human Body 2 ($n = 2$)	0,895	0,636	**0,708**	**0,584**	**0,944**	0,525	0,636	0,504	**0,773**	**0,662**	0,668	0,536
IBM test 2 ($n = 6$)	0,845	0,714	**0,721**	**0,600**	**0,900**	0,664	0,706	0,595	**0,735**	**0,733**	0,651	0,514
People & Foliage ($n = 2$)	0,987	**0,767**	**0,841**	**0,756**	**0,992**	0,527	0,647	0,523	0,990	0,712	0,807	0,705
Snellen ($n = 2$)	0,940	**0,945**	**0,933**	**0,889**	**0,983**	0,610	0,723	0,600	0,950	0,801	0,855	0,762
Moyenne	0,792	0,644	**0,649**	**0,546**	**0,830**	0,549	0,584	0,468	0,654	**0,744**	0,601	0,489

Table 7.2. Scores (P, R, F et J) obtenus par trois méthodes dont la nôtre, dénotée DO, sur treize séquences de la base de données SBI 2015. La dernière ligne donne la moyenne sur l'ensemble des treize séquences.

Séquence	DO				EC-LGOF				SOFT			
	P	R	F	J	P	R	F	J	P	R	F	J
#1 ($n = 2$)	0,932	**0,744**	**0,819**	**0,703**	**0,987**	0,584	0,719	0,579	0,911	0,743	0,803	0,685
#2 ($n = 2$)	0,849	0,724	**0,780**	**0,641**	**0,941**	0,396	0,556	0,386	0,723	**0,812**	0,762	0,620
#3 ($n = 2$)	0,912	0,555	0,688	0,526	**0,965**	0,387	0,550	0,381	0,620	**0,810**	**0,695**	**0,534**
#4 ($n = 3$)	0,742	0,593	**0,649**	**0,484**	**0,780**	0,508	0,594	0,426	0,576	**0,689**	0,614	0,446
#5 ($n = 4$)	0,584	0,566	0,562	0,396	**0,809**	0,569	**0,610**	**0,456**	0,079	**0,962**	0,145	0,079
#6 ($n = 2$)	**0,678**	0,799	**0,729**	**0,577**	0,204	**0,983**	0,333	0,203	0,153	0,932	0,260	0,151
#7 ($n = 3$)	0,846	0,951	0,883	0,805	**0,961**	0,884	**0,916**	**0,850**	0,509	**0,960**	0,619	0,493
#8 ($n = 2$)	0,698	0,536	**0,603**	**0,437**	**0,705**	0,516	0,589	0,425	0,365	**0,745**	0,457	0,306
#9 ($n = 2$)	**0,731**	0,461	**0,521**	**0,396**	0,623	0,495	0,465	0,351	0,455	**0,775**	0,502	0,378
#10 ($n = 2$)	0,869	0,303	0,445	0,289	**0,925**	0,149	0,247	0,147	0,567	**0,801**	**0,637**	**0,474**
Moyenne	**0,675**	0,578	**0,586**	**0,448**	0,669	0,620	0,538	0,413	0,408	**0,851**	0,470	0,348

Table 7.3. Scores (P, R, F et J) obtenus par trois méthodes dont la nôtre, dénotée DO, sur dix séquences de la base de données du projet MIVAO. La dernière ligne donne la moyenne sur l'ensemble des 90 séquences. Le nom des séquences est anonymisée pour des raisons de confidentialité.

8 Conclusion

Devant l'augmentation de la fréquentation des domaines skiables ces dernières décennies, la sécurité de leurs clients est devenue la préoccupation majeure des gérants de stations. C'est particulièrement le cas sur les remontées mécaniques où 92% des accidents sont liés au seul comportement humain. Dans ce contexte, la start-up BLUECIME, les laboratoires GIPSA-LAB et HUBERT CURIEN, ainsi que le groupe SOFIVAL se sont unis autour du projet MIVAO dont le but est de construire une intelligence artificielle capable de détecter, voire d'anticiper, une situation dangereuse à bord de véhicules d'un télésiège. Cette thèse s'est concentrée sur l'annotation automatique de vidéos de rémontées mécaniques, et ce en prenant en compte les différentes variabilités associées (taille du véhicule, géométrie du télésiège, illumination pour n'en citer que quelques unes).

En Partie I, nous avons présenté une technique de classification permettant de distinguer les images représentant des véhicules vides de ceux qui comptent des passagers. Cette information préliminaire est en effet essentielle pour la détection d'un danger potentiel. La méthode proposée est basée sur les analyses discriminantes linéaire et quadratique et sur des caractéristiques, extraites des images originales, interprétables physiquement, reposant sur des éléments statistiques et liés au traitement de l'image classique. Une classification plus robuste et efficace est ensuite obtenue via l'inclusion d'informations *a priori* sur le système étudié. La technique proposée présente des résultats compétitifs comparés à d'autres plus complexes et un fonctionnement en temps-réel.

Nous avons détaillé en Partie II un processus permettant de dénombrer les passagers de chaque véhicule de la manière la moins supervisée possible. La méthode présentée se décompose en trois étapes :

1. **Réduction de dimension** pour ne garder que l'information nécessaire à l'application.
2. **Partitionnement de données** afin de regrouper les passages de véhicules similaires.
3. **Transduction** dans le but d'inclure l'information exacte du nombre de passagers.

La méthode présente des performances encourageantes en termes de partitionnement, également en matière de classification du nombre de passagers présents sur chaque véhicule, grâce à la dernière étape de transduction qui permet de plus une certaine robustesse. En particulier, nous avons proposé dans cette partie deux algorithmes de partitionnement de données. Le premier concilie clusterings par mélange Gaussien et spectral. Le deuxième généralise la méthode DBSCAN en remplaçant les voisinages sphériques par des ellipsoïdes, ce qui permet une adaptation aux données au niveau local.

Quant à la Partie III, elle se consacre à l'extraction des images unitaires à partir de celles issues de la caméra, autrement dit à la détection des véhicules et à leur suivi dans la scène. Nous avons d'abord proposé une méthode, basée sur l'analyse fréquentielle d'une vidéo, permettant de déterminer automatiquement la durée d'un passage. Un algorithme de détection d'objets en mouvement, reposant sur l'étude de la déformation temporelle du flux optique, a également été présenté. Ce dernier montre de bonnes performances par rapport à deux autres méthodes de l'état de l'art, en termes de PPC et F-mesure.

Nous souhaitons également souligner que nous nous sommes attachés dans cette thèse à proposer des méthodes génériques, c'est-à-dire qui peuvent évidemment être appliquées à la problématique de la sécurité des passagers de remontées mécaniques, mais qui peuvent aussi être utilisées sans souçi dans un autre contexte.

Les perspectives en ce qui concerne les travaux présentés dans ce manuscrit sont nombreuses, notamment car la problématique générale apportée par le projet MIVAO est nouvelle.

Premièrement, bien que les résultats soient déjà satisfaisants, la robustesse de la méthode de classification vide/non-vide proposée en Partie I peut être améliorée par empilement de modèles (*predictor stacking* en anglais). L'idée consiste à utiliser différents modèles pour une même image, puis à choisir ceux qui aboutissent à la plus grande confiance (traduite par la distance à la fonction de séparation par exemple).

En ce qui concerne la Partie II, les pistes d'amélioration se concentrent principalement sur la proposition d'une distance efficace entre images unitaires. D'abord, l'utilisation d'un autre type de distance, à la place de l'IMage Euclidean Distance (IMED), pourrait être plus appropriée. De plus, nous pensons que l'information de couleurs serait bénéfique afin de différencier les passagers du véhicule et de l'arrière-plan.

L'algorithme Gaussian Spectral Clustering (GSC) proposé en section 5.4 est également sujet à améliorations. Au-delà de celles déjà explorées, la principale source de progrès potentiel se situe au niveau de la modélisation par mélange Gaussien : d'une part, l'aspect aléatoire, notamment à l'initialisation, est à réduire. D'autre part, rappelons que la détermination du mélange Gaussien qui représente au mieux un jeu de données spécifique s'effectue via la minimisation du Bayesian Information Criterion (BIC) : ce processus n'est pas optimal car la recherche est exhaustive, c'est-à-dire réalisée sur un grand nombre de modèles possibles. Quant à cette recherche, le travail à apporter à l'algorithme EN-DBSCAN, proposé en section 5.3, se concentrera sur la gestion des données en grandes dimensions. En particulier, le choix des poids utilisés (inverse du carré de la distance Euclidienne comme actuellement) est crucial.

Les perspectives liées à la Partie III sont la modélisation complète et automatique de la trajectoire des véhicules de n'importe quelle remontée. De manière naïve, la concaténation des images obtenues dans ce chapitre (*i.e.* la séparation premier-plan/arrière-plan) pourrait permettre une telle modélisation. Par la suite, il serait possible de localiser le véhicule d'intéret à chaque instant. Une autre application pourrait être la création, de manière automatique, du masque du véhicule à détecter dans l'image, tout en prenant en compte son évolution au cours du temps (due à la perspective).

Sur une note plus générale, on pourrait imaginer à moyen-terme pouvoir appliquer les différents travaux présentés dans ce manuscrit à d'autres types de transports guidés (par exemple téléphériques, tramways, chemins de fer, *etc*.), du fait de leur similarité avec les remontées mécaniques. De plus, nous avons insisté lors de cette thèse sur l'aspect non-supervisé des méthodes proposées. En effet, nous pensons que ce dernier prendra de manière générale davantage de place dans un futur proche, étant donné la quantité de données qui prolifèrent grâce à Internet et l'effort incommensurable à produire pour les annoter.

Bibliographie

[Agr+98] Rakesh AGRAWAL, Johannes GEHRKE, Dimitrios GUNOPULOS et Prabhakar RAGHAVAN. “Automatic subspace clustering of high dimensional data for data mining applications”. In : *SIGMOD*. 1998, p. 94–105 (cf. p. 69).

[Aiz+64] M. A. AIZERMAN, E. BRAVERMAN et L. I. ROZONER. “Theoretical foundations of the potential function method in pattern recognition learning”. In : *Automation and Remonte Control* 25.6 (juin 1964), p. 917–936 (cf. p. 48).

[Aka74] Hirotugu AKAIKE. “A new look at the statistical model identification”. In : *IEEE Transactions on Automatic Control* 19.6 (déc. 1974), p. 716–723 (cf. p. 90).

[AS+16] Ma’moun AL-SMADI, Khairi ABDULRAHIM et Rosalina Abdul SALAM. “Cumulative frame differencing for urban vehicle detection”. In : *First International Workshop on Pattern Recognition*. Juil. 2016 (cf. p. 140).

[Ank+99] Mihael ANKERST, Markus M. BREUNIG, Hans-Peter KRIEGEL et Joerg SANDER. “OPTICS : Ordering Points to Identify the Clustering Structure”. In : *ACM SIGMOD*. T. 28. 2. Juin 1999, p. 49–60 (cf. p. 69).

[AE01] George ARIMOND et Abdulaziz ELFESSI. “A clustering method for categorical data in tourism market segmentation research”. In : *Journal of Travel Research* 39.4 (mai 2001), p. 391–397 (cf. p. 63).

[AV07] David ARTHUR et Sergei VASSILVITSKII. “K-means++ : the advantages of careful seeding”. In : *ACM-SIAM symposium on discrete algorithms*. Jan. 2007 (cf. p. 66, 99, 104).

[Art15] Emil ARTIN. *The Gamma Function*. Sous la dir. de Dover PUBLICATIONS. Dover Publications, 2015 (cf. p. 191).

[AMN13] Sepehr ASLANI et Homayoun MAHDAVI-NASAB. “Optical flow based moving object detection and tracking for traffic surveillance”. In : *International Journal of Electrical and Computer Engineering* 7.9 (2013), p. 1252–1256 (cf. p. 141).

[Aue+02] Peter AUER, Nicolo CESA-BIANCHI et Paul FISCHER. “Finite-time Analysis of the Multiarmed Bandit Problem”. In : *Machine Learning* 47 (2002), p. 235–256 (cf. p. 108).

[BJ06] Francis R. BACH et Michael I. JORDAN. “Learning Spectral Clustering, With Application To Speech Separation”. In : *Journal of Machine Learning Research* 7 (oct. 2006), p. 1963–2001 (cf. p. 71).

[Bas+17] Kévin BASCOL, Rémi EMONET, Elisa FROMONT et Raluca DEBUSSCHERE. “Improving chairlift security with deep learning”. In : *International Symposium on Intelligent Data Analysis*. T. 10584. Oct. 2017, p. 1–13 (cf. p. 21, 22, 26).

[Bel+97] Peter N. BELHUMEUR, Joao P. HESPANHA et David J. KRIEGMAN. “Eigenfaces vs. Fisherfaces : Recognition Using Class Specific Linear Projection”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (juil. 1997), p. 711–720 (cf. p. 47).

[BN03] Mikhail BELKIN et Partha NIYOGI. “Laplacian eigenmaps for dimensionality reduction and data representation”. In : *Neural Computation* 15.6 (juin 2003), p. 1373–1396 (cf. p. 48, 51, 52).

[Bel03] Richard Ernest BELLMAN. *Dynamic Programming*. Sous la dir. de Courier CORPORATION. Dover Publications, 2003 (cf. p. 42).

[Bez81] James C. BEZDEK. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Sous la dir. de SPRINGER. Plenum, jan. 1981 (cf. p. 109).

[Bha43] A. BHATTACHARYYA. “On a Measure of Divergence Between Two Statistical Populations Defined by their Probability Distributions”. In : *Bulletin of the Calcutta Mathematical Society* (1943), p. 99–109 (cf. p. 97).

[Blu60] L. E. BLUMENSON. “A derivation of n-dimensional spherical coordinates”. In : *The American Mathematical Monthly* 67.1 (jan. 1960), p. 63–66 (cf. p. 75).

[BB16] Johannes BLÖMER et Kathrin BUJNA. “Simple Methods for Initializing the EM Algorithm for Gaussian Mixture Models”. In : *arXiv* (août 2016) (cf. p. 107).

[BG05] I. BORG et P. J. F. GROENEN. *Modern Multidimensional Scaling - Theory and Applications*. Sous la dir. de SPRINGER. 2. Springer-Verlag New York, 2005 (cf. p. 48).

[Bra+96] Paul S. BRADLEY, Olvi L. MANGASARIAN et Nick STREET. “Clustering via Concave Minimization”. In : *Advances in neural information processing systems* (jan. 1996), p. 368–374 (cf. p. 66).

[Bru09] Bruno BRUNELLI. *Template Matching Techniques in Computer Vision : Theory and Practice*. Sous la dir. de WILEY. Wiley, mar. 2009 (cf. p. 136).

[But+12] Daniel J. BUTLER, Jonas WULFF, Garrett B. STANLEY et Michael J. BLACK. “A naturalistic open source movie for optical flow evaluation”. In : *European Conference on Computer Vision*. T. 7577. 2012, p. 611–625 (cf. p. 143, 145).

[Bö+04] Christian BÖHM, Karin KAILING, Peer KRÖGER et Arthur ZIMEK. “Computing Clusters of Correlation Connected Objects”. In : *International Conference on Management of Data (SIGMOD)*. Juin 2004, p. 455–466 (cf. p. 70).

[Cae+19] Sergi CAELLES, Jordi PONT-TUSET, Federico PERAZZI et al. “The 2019 DAVIS Challenge on VOS : Unsupervised Multi-Object Segmentation”. In : *arXiv* (mai 2019) (cf. p. 144, 149–151).

[Cam+13] Ricardo J.G.B. CAMPELLO, Davoud MOULAVI et Joerg SANDER. “Density-based clustering based on hierarchical density estimates”. In : *Advances in Knowledge Discovery and Data Mining*. T. 7819. Springer, avr. 2013, p. 160–172 (cf. p. 69).

[CG92] Gilles CELEUX et Gérard GOVAERT. “A classification EM algorithm for clustering and two stochastic versions”. In : *Computational Statistics and Data Analysis* 14.3 (oct. 1992), p. 315–332 (cf. p. 94).

[CD85] Gérard CELEUX et J. DIEBOLT. “The SEM algorithm : a probabilistic teacher algorithm derived from the em algorithm for the mixture problem”. In : *Computational Statistics Quarterly* 2 (1985), p. 73–82 (cf. p. 94).

[Chi92] Nancy CHINCHOR. “MUC-4 evaluation metrics”. In : *Message Understanding Conference*. Juin 1992, p. 22–29 (cf. p. 19, 151).

[Cho+19] Jaechan CHO, Yongchul JUNG, Dong-Sun KIM, Seongjoo LEE et Yunho JUNG. “Moving Object Detection Based on Optical Flow Estimation and a Gaussian Mixture Model for Advanced Driver Assistance Systems”. In : *Sensors* 19.14 (juil. 2019) (cf. p. 141).

[CM90] Alexandre J. CHORIN et J.E. MARSDEN. *A mathematical introduction to fluid mechanics*. Sous la dir. de SPRINGER. T. 4. Texts in Applied Mathematics 2. Springer, 1990 (cf. p. 141, 142).

[CL06] Ronald R. COIFMAN et Stéphane LAFON. “Diffusion maps”. In : *Applied and Computational Harmonic Analysis* 21.1 (juil. 2006), p. 5–30 (cf. p. 48, 53, 55).

[CV95] Corinna CORTES et Vladimir VAPNIK. “Support-vector networks”. In : *Machine learning* 20.3 (sept. 1995), p. 273–297 (cf. p. 27, 189).

[Cos+16] Serhan COSAR, Guiseppe DONATELLIO, Vania BOGORNY et al. “Toward Abnormal Trajectory and Event Detection in Video Surveillance”. In : *IEEE Transactions on Circuits and Systems for Video Technology* 27.3 (juil. 2016), p. 683–695 (cf. p. 140).

[Cra02] Jan Salomon CRAMER. *The origins of logistic regression*. Rapp. tech. Tinbergen institute, nov. 2002 (cf. p. 27).

[CR+07] Frédérique CRÉTÉ-ROFFET, Thierry DOLMIERE, Patricia LADRET et Marina NICOLAS. “The Blur Effect : Perception and Estimation with a New No-Reference Perceptual Blur Metric”. In : *Proceedings of SPIE - The International Society for Optical Engineering* 6492 (fév. 2007) (cf. p. 17).

[DT05] Navneet DALAL et Bill TRIGGS. “Histograms of oriented gradients for human detection”. In : *Computer Vision and Pattern Recognition (CVPR)*. 2005 (cf. p. 29).

[Dao+20] Vinh-Loc DAO, Cécile BOTHOREL et Philippe LENCA. “Community structure : A comparative evaluation of community detection methods”. In : *Network Science* 8.1 (jan. 2020), p. 1–41 (cf. p. 63).

[Def77] D. DEFAYS. “An efficient algorithm for a complete link method”. In : *The Computer Journal* 20.4 (jan. 1977), p. 364–366 (cf. p. 65).

[Dem+77] A. P. DEMPSTER, N. M. LAIRD et D. B. RUBIN. “Maximum Likelihood from Incomplete Data via the EM Algorithm”. In : *Journal of the Royal Statistical Society* 39.1 (1977), p. 1–38 (cf. p. 92).

[DM01] Jean DEZERT et Christian MUSSO. *An Efficient Method for Generating Points Uniformly Distributed in Hyperellipsoids*. 2001 (cf. p. 195).

[Dha+15] Nameirakpam DHANACHANDRA, Khumanthem MANGLEM et Yambem Jina CHANU. "Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm". In : *Procedia Computer Science* 54 (2015), p. 764–771 (cf. p. 67).

[Dhi+04] Inderjit S. DHILLON, Yuqiang GUAN et Brian KULIS. "Kernel k-means : spectral clustering and normalized cuts". In : *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Août 2004, p. 551–556 (cf. p. 66).

[DG03] David DONOHO et Carrie GRIMES. "Hessian Eigenmaps : Locally linear embedding techniques for high-dimensional data". In : *Proceedings of the National Academy of Science* 100.10 (juin 2003), p. 5591–5596 (cf. p. 51).

[Dun73] J.C. DUNN. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In : *Journal of Cybernetics and Systems* 3.3 (sept. 1973), p. 32–57 (cf. p. 66, 109).

[Est+97] Martin ESTER, Kriegel HANS-PETER, Jörg SANDER et Xiaowei XU. "A density-based algorithm for discovering clusters in large spatial databases with noise". In : *International conference on knowledge discovery and data mining*. Déc. 1997, p. 226–231 (cf. p. 7, 67, 68, 83, 105).

[EC02] Vladimir ESTIVILL-CASTRO. "Why so many clustering algorithms : a position paper". In : *ACM SIGKDD Explorations Newsletter* 4.1 (juin 2002), p. 65–75 (cf. p. 62).

[Fad+17] Marzieh FADAEE, Arianna BISAZZA et Christof MONZ. "Data Augmentation for Low-Resource Neural Machine Translation". In : *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Juil. 2017, p. 567–573 (cf. p. 118).

[FJ02] M.A.T. FIGUEIREDO et A.K. JAIN. "Unsupervised learning of finite mixture models". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3 (mar. 2002), p. 381 –396 (cf. p. 90).

[Fis36a] Ronald A. FISHER. *Iris flower dataset*. 1936 (cf. p. 14).

[Fis36b] Ronald A. FISHER. "The use of multiple measurements in taxonomic problems". In : *Annals of Eugenics* 7.7 (sept. 1936), p. 179–188 (cf. p. 13, 46).

[FM83] Elizabeth B. FOWLKES et Colin L. MALLOWS. "A method for comparing two hierarchical clusterings". In : *Journal of the American Statistical Association* 78.383 (sept. 1983), p. 553–569 (cf. p. 85, 101, 123).

[Fri+08] Jerome FRIEDMAN, Trevor HASTIE et Robert TIBSHIRANI. "Sparse inverse covariance estimation with the graphical lasso". In : *Biostatistics* 9.3 (juil. 2008), p. 432–441 (cf. p. 78, 94).

[FH75] Keinosuke FUKUNAGA et Larry D. HOSTETLER. "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition". In : *IEEE Transactions on Information Theory* 21.1 (jan. 1975), p. 32–40 (cf. p. 69).

[Gam+19] Ferran P. GAMONAL, C. BALLESTER, Gloria HARO et Palomares Roberto P. MEINHARDT-LLOPIS Enric. “An analysis and speedup of the FALDOI method for Optical Flow estimation”. In : *Image Processing On Line* 9 (2019), p. 94–123 (cf. p. 141).

[GK78] Donald E. GUSTAFSON et William C. KESSEL. “Fuzzy Clustering with a Fuzzy Covariance Matrix”. In : *Proceedings of the IEEE Conference on Decision and Control*. T. 17. Fév. 1978, p. 761–766 (cf. p. 71).

[HS88] Christopher G. HARRIS et Mike STEPHENS. “A Combined Corner and Edge Detector”. In : *Alvey Vision Conference*. 1988, p. 147–151 (cf. p. 29).

[HW79] John HARTIGAN et Manchek WONG. “Algorithm AS 136 : A K-Means Clustering Algorithm”. In : *Applied Statistics* 28.1 (1979), p. 100–108 (cf. p. 66).

[Has+09] Trevor HASTIE, Robert TIBSHIRANI et Jerome FRIEDMAN. *The elements of statistical learning*. Sous la dir. de SPRINGER. Springer, 2009 (cf. p. 15).

[HM09] Omnia Ossama HODA et Hoda MOKHTAR. “Similarity search in moving object trajectories”. In : *Proceedings of the 15th International Conference on Management of Data*. Déc. 2009 (cf. p. 140).

[Hot33] Harold HOTELLING. “Analysis of a complex of statistical variables into principal components”. In : *Journal of Educational Psychology* 24.6 (1933), p. 417–441 (cf. p. 45).

[Hua+19] Junjie HUANG, Wei ZOU, Zhu ZHENG et Jiagang ZHU. “An efficient optical flow based motion detection method for non-stationary scenes”. In : *Chinese Control And Decision Conference (CCDC)*. 2019 (cf. p. 141).

[Kal+15] Kiran KALE, Sushant PAWAR et Pravin DHULEKAR. “Moving object tracking using optical flow and motion vector estimation”. In : *International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*. Sept. 2015 (cf. p. 141).

[KR87] Leonard KAUFMAN et Peter J. ROUSSEEUW. “Clustering by Means of Medoids”. In : *Delft University of Technology : reports of the Faculty of Technical Mathematics and Informatics* 87.3 (1987) (cf. p. 66).

[KR05] Leonard KAUFMAN et Peter J. ROUSSEEUW. *Finding Groups in Data : An Introduction to Cluster Analysis*. Sous la dir. de WILEY. Wiley Interscience, 2005 (cf. p. 64).

[Ken64] E.S. KENNEY John Francis ans Keeping. *Mathematics of Statistics*. Sous la dir. de Van NOSTRAND. Van Nostrand, 1964 (cf. p. 45).

[Ker00] Christine KERIBIN. “Consistent estimation of the order of mixture models”. In : *Sankhyā : The Indian Journal of Statistics* 62.1 (fév. 2000), p. 49–66 (cf. p. 95).

[Kha+14] Samina KHALID, Tehmina KHALIL et Shamila NASREEN. “A survey of feature selection and feature extraction techniques in machine learning”. In : *Science and Information Conference*. 2014 (cf. p. 43).

[KI86] J. KITTLER et J. ILLINGWORTH. “Minimum error thresholding”. In : *Pattern Recognition* 19.1 (1986), p. 41–47 (cf. p. 149).

[Kra91] Mark A. KRAMER. “Nonlinear principal component analysis using autoassociative neural networks”. In : *American Institute of Chemical Engineers* 37.2 (fév. 1991), p. 233–243 (cf. p. 48).

[Kri+12] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E. HINTON. “Imagenet classification with deep convolutional neural networks”. In : *Advances in neural information processing systems*. T. 25. 2. Jan. 2012, p. 1097–1105 (cf. p. 118).

[Kru64a] J. B. KRUSKAL. “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”. In : *Psychometrika* 29.1 (mar. 1964), p. 1–27 (cf. p. 48).

[Kru64b] J. B. KRUSKAL. “Nonmetric multidimensional scaling : a numerical method”. In : *Psychometrika* 29 (juin 1964), p. 115–129 (cf. p. 48).

[Lai+12] Yu-Ren LAI, Kuo-Liang CHUNG, Guei-Yin LIN et Chyou-Hwa CHEN. “Gaussian mixture modeling of histograms for contrast enhancement”. In : *Expert Systems with Applications* 39.8 (juin 2012), p. 6720–6728 (cf. p. 92).

[LR05] Erich L. LEHMANN et Joseph P. ROMANO. *Testing statistical hypotheses*. Sous la dir. de SPRINGER. 3. Springer-Verlag New York, 2005 (cf. p. 191).

[Ler92] Brian G. LEROUX. “Consistent estimation of a mixing distribution”. In : *Annals of Statistics* 20.3 (1992), p. 1350–1360 (cf. p. 95).

[LL09] Yan LI et Lei LI. “A Novel Split and Merge EM Algorithm for Gaussian Mixture Model”. In : *International Conference on Natural Computation*. IEEE, août 2009, p. 479–483 (cf. p. 90).

[Lio+16] Sze Teng LIONG, John SEE, Raphael C.W. PHAN et al. “Spontaneous subtle expression detection and recognition based on facial strain”. In : *Signal Processing Image Communication* 47 (sept. 2016), p. 170–182 (cf. p. 146).

[Liu+13a] Xin LIU, Duygu TOSEN et Schuff Norbert WEINER Michael W. “Locally linear embedding (LLE) for MRI based Alzheimer’s disease classification”. In : *NeuroImage* 83 (déc. 2013), p. 148–157 (cf. p. 50).

[Liu+13b] Yanchi LIU, Zhongmou LI, Hui XIONG et al. “Understanding and enhancement of internal clustering validation measures”. In : *IEEE Transactions on Cybernetics*. T. 43. 3. IEEE, juin 2013, p. 982–994 (cf. p. 100).

[Low04] David LOWE. “Distinctive Image Features from Scale-Invariant Keypoints”. In : *International Journal of Computer Vision* 60.2 (nov. 2004), p. 91–110 (cf. p. 29).

[Lux07] Ulrike von LUXBURG. “A Tutorial on Spectral Clustering”. In : *Statistics and Computing* 17.4 (2007), p. 395–416 (cf. p. 98).

[Maa14] Laurens van der MAATEN. “Accelerating t-SNE using Tree-Based Algorithms”. In : *Journal of Machine Learning Research* 15.96 (jan. 2014), p. 3221–3245 (cf. p. 48).

[MH08] Laurens van der MAATEN et Geoffrey HINTON. “Visualizing Data using t-SNE”. In : *Journak of Machine Learning Research* (2008) (cf. p. 48).

[MP15] Lucia MADDALENA et Alfredo PETROSSINO. “Towards Benchmarking Scene Background Initialization”. In : *International Conference on Image Analysis and Processing*. Août 2015, p. 469–476 (cf. p. 150).

[Mah36] P. C. MAHALANOBIS. “On the generalized distance in statistics”. In : *Proceedings of the National Institute of Sciences (Calcutta)* 2.1 (1936), p. 49–55 (cf. p. 71, 73, 76).

[Mai+16] Gengchen MAI, Krzysztof JANOWICZ, Song GAO et Yingjie HU. “ADCN : An Anisotropic Density-Based Clustering Algorithm”. In : *SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 58. Oct. 2016 (cf. p. 69).

[Mai+17] Gengchen MAI, Krzysztof JANOWICZ, Song GAO et Yingjie HU. “ADCN : An Anisotropic Density-Based Clustering Algorithm for Discovering Spatial Point Patterns with Noise”. In : *Transactions in GIS* 22.1 (oct. 2017) (cf. p. 69).

[Mar72] George MARSAGLIA. “Choosing a point from the surface of a sphere”. In : *The Annals of Mathematical Statistics* 43.2 (1972), p. 645–646 (cf. p. 191).

[Mat03] Yasuo MATSUYAMA. “The alpha-EM algorithm : surrogate likelihood maximization using /spl alpha/-logarithmic information measures”. In : *IEEE Transactions on Information Theory* 49 (jan. 2003), p. 692–706 (cf. p. 94).

[McI+18] Leland MCINNES, Jon HEALY et James MELVILLE. “UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction”. In : *arXiv* (2018) (cf. p. 48).

[MR14] Geoffrey J. MCLACHLAN et Suren RATHNAYAKE. “On the number of components in a Gaussian mixture model”. In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 4.5 (sept. 2014), p. 341–355 (cf. p. 90, 95).

[Mic27] Albert Abraham MICHELSON. *Studies in optics*. Sous la dir. de DOVER. The university of Chicago press, 1927 (cf. p. 17).

[Muz+20] Julien MUZEAU, Maria OLIVER-PARERA, Patricia LADRET et Pascal BERTOLINO. “Combining Mixture Models and Spectral Clustering for Data Partitioning”. In : *International Conference on Image Analysis and Recognition*. T. 12132. Springer, juin 2020, p. 63–75 (cf. p. 8, 110).

[Muz+19] Julien MUZEAU, Patricia LADRET et Pascal BERTOLINO. “Linear classification of chairlift images for presence analysis”. In : *Fourteenth International Conference on Quality Control by Artificial Vision*. Juil. 2019 (cf. p. 7, 30).

[Ng+01] Andrew Y. NG, Michael I. JORDAN et Yair WEISS. “On spectral clustering : analysis and an algorithm”. In : *Advances in Neural Information Processing Systems*. MIT Press, 2001, p. 849–856 (cf. p. 70, 98, 104).

[Oli+00] Nuria M. OLIVER, Barbara ROSARIO et Alex Paul PENTLAND. “A Bayesian computer vision system for modeling human interactions”. In : *IEEE transactions on pattern analysis and machine intelligence*. T. 22. 8. Août 2000 (cf. p. 45).

[Ots79] Nobuyuki OTSU. “A Threshold Selection Method from Gray-Level Histograms”. In : *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (jan. 1979), p. 62–66 (cf. p. 149).

[Pal+17] Roberto P. PALOMARES, Enric MEINHARDT-LLOPIS, Coloma BALLESTER et Gloria HARO. “Faldoi : A new minimization strategy for large displacement variational optical flow”. In : *Journal of Mathematical Imaging and Vision* 58 (2017), p. 27–46 (cf. p. 141).

[Par62] Emmanuel PARZEN. “On estimation of a probability density function and mode”. In : *Annals of Mathematical Statistics* 33.3 (1962), p. 1065–1076 (cf. p. 69).

[PP12] Chirag I. PATEL et Ripta PATEL. “Contour based object tracking”. In : *International Journal of Computer and Electrical Engineering* 4.4 (août 2012), p. 525–528 (cf. p. 141).

[PP14] Milin P. PATEL et Shankar K. PARMAR. “Moving object detection with moving background using optic flow”. In : *International Conference on Recent Advances and Innovations in Engineering*. IEEE, mai 2014 (cf. p. 141, 149).

[Pat05] Vasile PATRASCU. “A Generalization of Gustafson-Kessel Algorithm Using a New Constraint Parameter”. In : *Conference of the European Society for Fuzzy Logic and Technology*. 2005 (cf. p. 71).

[Pau+17] Nihal PAUL, Ashish SINGH, Abhishek MIDYA, Partha Pratim ROY et Debi Prosad DOGRA. “Moving object detection using modified temporal differencing and local fuzzy thresholding”. In : *Journal of Supercomputing* 73.3 (mar. 2017), p. 1120–1139 (cf. p. 140).

[Pea01] Karl PEARSON. “On lines and planes of closest fit to systems of points in space”. In : *Philosophical Magazine* 2.6 (1901), p. 559–572 (cf. p. 44, 45).

[Pea05] Karl PEARSON. “The Problem of the Random Walk”. In : *Nature* 72.294 (juil. 1905) (cf. p. 54).

[PM00] Dan PELLEG et Andrew MOORE. “X-means : Extending K-means with Efficient Estimation of the Number of Clusters”. In : *In Proceedings of the 17th International Conf. on Machine Learning*. 2000, p. 727–734 (cf. p. 100).

[PM18] Wilmar Hernandez PERDOMO et Alfredo MENDEZ. *Application of Principal Component Analysis to Image Compression*. Sous la dir. d’INTECHOPEN. IntechOpen, 2018 (cf. p. 45).

[Rad+14] Syaimaa Solelah Mohd RADZI, Shahrul Nizam YAAKOB, Zulaikha KADIM et Hon Hock WOON. “Extraction of Moving Objects Using Frame Differencing, Ghost and Shadow Removal”. In : *International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*. Jan. 2014 (cf. p. 140).

[Rem+01] Maido REMM, Christian E.V. STORM et Erik L.L. SONHAMMER. “Automatic clustering of orthologs and in-paralogs from pairwise species comparisons”. In : *Journal of Molecular Biology* 314.5 (déc. 2001), p. 1041–1052 (cf. p. 63).

[RW97] Kathryn ROEDER et Larry WASSERMAN. “Practical Bayesian density estimation using mixtures of normals”. In : *Journal of the American Statistical Association* 92.439 (1997), p. 894–902 (cf. p. 95).

[RS00] Sam T. ROWEIS et Lawrence K. SAUL. “Nonlinear dimensionality reduction by locally linear embedding”. In : *Science* 290.5500 (déc. 2000), p. 2323–2326 (cf. p. 48).

[RG20] Sujoy Madhab ROY et Ashish GHOSH. “Foreground Segmentation Using Adaptive 3 Phase Background Model”. In : *IEEE Transactions on Intelligent Transportation Systems* 21.6 (juin 2020), p. 2287–2296 (cf. p. 141).

[Rua+11] Lingyan RUAN, Ming YUAN et Hui ZOU. “Regularized Parameter Estimation in High-Dimensional Gaussian Mixture Models”. In : *Neural Computation* 23.6 (mar. 2011), p. 1605–1622 (cf. p. 94).

[Rum+86] David E. RUMELHART, Geoffrey E. HINTON et Ronald J. WILLIAMS. “Learning representations by back-propagating errors”. In : *Nature* 323 (oct. 1986), p. 533–536 (cf. p. 52).

[San+13] Javier SANCHEZ, Enric MEINHARDT-LLOPIS et Gabriele FACCIOLO. “TV-L1 Optical Flow Estimation”. In : *Image Processing On Line* 3 (juil. 2013), p. 137–150 (cf. p. 141, 143, 152).

[San+98] Jörg SANDER, Martin ESTER, Hans-Peter KRIEGEL et Xiaowei XU. “Density-Based Clustering in Spatial Databases : The Algorithm GDBSCAN and Its Applications”. In : *Data Mining and Knowledge Discovery* (juin 1998), p. 169–194 (cf. p. 69, 86).

[SR01] Lawrence K. SAUL et Sam T. ROWEIS. “An introduction to locally linear embedding”. In : *Journal of Machine Learning Research* (jan. 2001) (cf. p. 48, 50).

[Sax+17] Amit SAXENA, Mukesh PRASAD, Akshansh GUPTA et al. “A review of clustering techniques and developments”. In : *Neurocomputing* 267 (déc. 2017), p. 664–681 (cf. p. 64).

[SG18] Erich SCHUBERT et Michael GERTZ. “Improving the Cluster Structure Extracted from OPTICS Plots”. In : *Lernen, Wissen, Daten, Analysen (LWDA)*. T. 2191. 2018, p. 318–329 (cf. p. 69).

[Sch+17] Erich SCHUBERT, Jörg SANDER, Martin ESTER, Hans-Peter KRIEGEL et Xiaowei XU. “DBSCAN Revisited, Revisited : Why and How You Should (Still) Use DBSCAN”. In : *ACM Transactions on Database Systems* 42.3 (juil. 2017), p. 1–21 (cf. p. 67, 69, 105).

[Sch78] Gideon SCHWARZ. “Estimating the Dimension of a Model”. In : *Annals of Statistics* 6.2 (1978), p. 461–464 (cf. p. 90, 95).

[Sch+98] Bernhard SCHÖLKOPF, Alexander SMOLA et Klaus-Robert MÜLLER. “Nonlinear Component Analysis as a Kernel Eigenvalue Problem”. In : *Neural Computation* 10.5 (juil. 1998), p. 1299–1319 (cf. p. 48, 49).

[Scu10] David SCULLEY. “Web-scale k-means clustering”. In : *Proceedings of the 19th international conference on World Wide Web*. Jan. 2010, p. 1177–1178 (cf. p. 66).

[SM00] Jianbo SHI et Jitendra MALIK. “Normalized cuts and image segmentation”. In : *IEEE transactions on Pattern Analysis and Machine Intelligence* 22.8 (août 2000), p. 888–905 (cf. p. 70).

[SK12] Subhash SHINDE et Uday KULKARNI. “Hybrid personalized recommender system using centering-bunching based clustering algorithm”. In : *Expert Systems with Applications* 39.1 (jan. 2012), p. 1381–1387 (cf. p. 63).

[ST07] Romy SHIODA et Levent TUNÇEL. “Clustering via minimum volume ellipsoids”. In : *Computational Optimization and Applications* 37.3 (avr. 2007), p. 247–295 (cf. p. 71).

[Shr13] Matthew SHREVE. “Automatic macro- and micro-facial expression spotting and applications”. Thèse de doct. University of South Florida, 2013 (cf. p. 142, 145).

[Sib73] R. SIBSON. “SLINK : an optimally efficient algorithm for the single-link cluster method”. In : *The Computer Journal* 16.1 (jan. 1973), p. 30–34 (cf. p. 65).

[Smi39] N. SMIRNOV. “On the estimation of the discrepancy between empirical curves of distribution for two independent samples”. In : *Bulletin. Math. Univ. Moscow* (1939) (cf. p. 88).

[SM58] Robert R. SOKAL et Charles D. MICHENER. “A Statistical Method for Evaluating Systematic Relationships”. In : *University of Kansas science bulletin* 38.22 (1958), p. 1409–1438 (cf. p. 65).

[SG99] Chris STAUFFER et W.E.L. GRIMSON. “Adaptive background mixture models for real-time tracking”. In : *Conference on Computer Vision and Pattern Recognition*. T. 2. 252. IEEE, fév. 1999 (cf. p. 141).

[SF08] Bing SUN et Jufu FENG. “A Fast Algorithm for Image Euclidean Distance”. In : *Chinese Conference on Pattern Recognition*. 2008 (cf. p. 116).

[Sya+12] Iwan SYARIF, Adam PRUGEL-BENNETT et Gary WILLS. “Unsupervised Clustering Approach for Network Anomaly Detection”. In : *International Conference on Networked Digital Technologies*. Springer, avr. 2012, p. 133–145 (cf. p. 63).

[Sze+15] Christian SZEGEDY, Wei LIU, Yangqing JIA et al. “Going deeper with convolutions”. In : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juin 2015 (cf. p. 118).

[Tal+13] Ronen TALMON, Israel COHEN, Sharon GANNOT et Ronald R. COIFMAN. “Diffusion maps for signal processing”. In : *IEEE Signal Processing Magazine* 30.4 (juin 2013), p. 75–86 (cf. p. 7, 48, 55, 116).

[Tan+19] Chao TANG, Huosheng HU, Miaohui ZHANG et al. “Real-time detection of moving objects in a video sequence by using data fusion algorithm”. In : *Transactions of the Institute of Measurement and Control* 41.3 (fév. 2019), p. 793–804 (cf. p. 141, 149).

[Ten+00] Joshua B. TENENBAUM, Vin de SILVA et John C. LANGFORD. “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. In : *Science* 290.5500 (déc. 2000), p. 2319–2323 (cf. p. 48).

[Tra+13] Thanh N. TRAN, Klaudia DRAB et Michal DASZYKOWSKI. “Revised DBSCAN algorithm to cluster data with dense adjacent clusters”. In : *Chemometrics and Intelligent Laboratory Systems* 120 (jan. 2013), p. 92–96 (cf. p. 69).

[Tsa+16] Yi-Hsuan TSAI, Ming-Hsuan YANG et Michael J. BLACK. “Video segmentation via object flow”. In : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juin 2016, p. 3899–3908 (cf. p. 141).

[TP91] Matthew TURK et Alex PENTLAND. “Face recognition using eigenfaces”. In : *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991 (cf. p. 45).

[Ued+00] Naonori UEDA, Ryohei NAKANO, Zoubin GHAHRAMANI et Geoffrey H. HINTON. “Split and Merge EM Algorithm for Improving Gaussian Mixture Density Estimates”. In : *Journal of VLSI Signal Processing* 26 (août 2000), p. 133–140 (cf. p. 90).

[Ush96] Akira USHIODA. “Hierarchical Clustering of Words and Application to NLP Tasks”. In : *Fourth Workshop on Very Large Corpora*. 1996 (cf. p. 66).

[VAR09] Stefan VAN AELST et Peter ROUSSEEUW. “Minimum Volume Ellipsoid”. In : *Wiley Interdisciplinary Reviews Computational Statistics* 1 (juil. 2009), p. 71–82 (cf. p. 71).

[VR19] Midhula VIJAYAN et Mohan RAMASUNDARAM. “A fast DGPSO-motion saliency map based moving object detection”. In : *Multimedia Tools and Applications* 78.1 (2019), p. 7055–7075 (cf. p. 141).

[Wal05] Christopher S. WALLACE. *Statistical and Inductive Inference by Minimum Message Length*. Sous la dir. de Springer-Verlag New YORK. Springer-Verlag New York, 2005 (cf. p. 90).

[Wan+05] Liwei WANG, Yan ZHANG et Jufu FENG. “On the Euclidean distance of images”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (sept. 2005), p. 1334–1339 (cf. p. 116).

[Wan12] Quan WANG. “Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models”. In : *arXiv* (juil. 2012) (cf. p. 49).

[Wan+97] Wei WANG, Jiong YANG et Richard R. MUNTZ. “STING : A statistical information grid approach to spatial data mining”. In : *International Conference on Very Large Data Bases*. Août 1997, p. 186–195 (cf. p. 69).

[War63] Joe H. Jr. WARD. “Hierarchical Grouping to Optimize an Objective Function”. In : *Journal of the American Statistical Association* 58.301 (mar. 1963), p. 236–244 (cf. p. 65).

[WC11] Andreas WEDEL et Daniel CREMERS. *Stereo Scene Flow for 3D Motion Analysis*. Sous la dir. de Springer-Verlag London LIMITED. Springer, London, 2011 (cf. p. 142).

[XT15] Dongkuan XU et Yingjie TIAN. “A Comprehensive Survey of Clustering Algorithms”. In : *Annals of Data Science* 2.2 (août 2015), p. 165–193 (cf. p. 64).

[YP05] M. YOKOYAMA et T. POGGIO. “A contour-based moving object detection and tracking”. In : *International Conference on Computer Communications and Networks*. Oct. 2005, p. 271–276 (cf. p. 141).

[Zac+07] Christopher ZACH, Thomas POCK et Horst BISCHOF. “A duality based approach for realtime TV-L1 optical flow”. In : *Proceedings of the 29th DAGM conference on Pattern recognition*. T. 4713. 2007, p. 214–223 (cf. p. 143).

[ZMP04] Lihi ZELNIK-MANOR et Pietro PERONA. “Self-Tuning Spectral Clustering”. In : *Advances in Neural Information Processing Systems*. Jan. 2004, p. 1601–1608 (cf. p. 71, 99).

[Zha+12] Hua ZHANG, Ruimin HU, Yimin WANG, Qingming LENG et Qiangguo CHEN. “A novel method of similarity search for moving object trajectories”. In : *International Conference on Automatic Control and Artificial Intelligence*. Mar. 2012, p. 235–238 (cf. p. 140).

[ZW06] Zhenyue ZHANG et Jing WANG. “MLLE : Modified Locally Linear Embedding Using Multiple Weights”. In : *Advances in neural information processing systems*. T. 19. Jan. 2006, p. 15993–1600 (cf. p. 51).

[ZZ02] Zhenyue ZHANG et Hongyuan ZHA. “Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment”. In : *SIAM Journal of Scientific Computing* 26.1 (2002), p. 313–338 (cf. p. 48).

[Zha+03] Zhihua ZHANG, Chibiao CHEN, Jian SUN et Kap Luk CHAN. “EM algorithms for Gaussian mixtures with split-and-merge operation”. In : *Pattern Recognition* 36.9 (sept. 2003), p. 1973–1983 (cf. p. 90).

[Zho+04] Dengyong ZHOU, Olivier BOUSQUET, Thomas Navin LAL, Jason WESTON et Bernhard SCHÖLKOPF. “Learning with Local and Global Consistency”. In : *Advances in neural information processing systems*. T. 16. 3. 2004, p. 321–238 (cf. p. 120, 127).

[ZG02] Xiaojin ZHU et Zoubin GHAHRAMANI. *Learning from Labeled and Unlabeled Data with Label Propagation*. Rapp. tech. Carnegie Mellon University, 2002 (cf. p. 121).

Table des figures

Liste des tableaux

A

Compléments théoriques sur l'analyse discriminante

Cette annexe donne les calculs complets menant aux résultats de la section 2.2 qui porte sur l'analyse discriminante.

En particulier, l'Équation 2.6 propose une expression explicite de l'ordonnée à l'origine w_o de l'hyperplan de discrimination optimale, paramétrisé jusqu'ici par son vecteur normal $\boldsymbol{w}$. Plaçons nous à nouveau dans la situation de la Figure 2.1b dans laquelle les données originales sont projetées sur l'hyperplan. On désigne par μ_j et σ_j les moyenne et écart-type de la classe j projetée. Le paramètre w_o peut se voir comme l'intersection des densités des probabilités jointes de chaque classe, ce qui donne :

$$p(w_o, \mathcal{C}_0) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} p(w_o, \mathcal{C}_1) \tag{A.1}$$

$$p(w_o|\mathcal{C}_0)\, p(\mathcal{C}_0) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} p(w_o|\mathcal{C}_1)\, p(\mathcal{C}_1) \tag{A.2}$$

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{w_o-\mu_0}{\sigma}\right)^2\right) \cdot \frac{n_0}{n} \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{w_o-\mu_1}{\sigma}\right)^2\right) \cdot \frac{n_1}{n} \tag{A.3}$$

$$-\frac{1}{2}\left(\frac{w_o-\mu_0}{\sigma}\right)^2 + \ln n_0 \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} -\frac{1}{2}\left(\frac{w_o-\mu_1}{\sigma}\right)^2 + \ln n_1 \tag{A.4}$$

$$-\frac{1}{2\sigma^2}\left[(w_o-\mu_0)^2 - (w_o-\mu_1)^2\right] \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} \ln\frac{n_1}{n_0} \tag{A.5}$$

$$(2w_o-\mu_0-\mu_1)(\mu_1-\mu_0) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} 2\sigma^2 \ln\frac{n_0}{n_1} \tag{A.6}$$

$$w_o \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} \frac{\mu_0+\mu_1}{2} + \frac{\sigma^2}{\mu_1-\mu_0}\ln\frac{n_0}{n_1} \tag{A.7}$$

On fait intervenir à l'Équation A.3 le fait que les deux probabilités conditionnelles suivent une distribution normale (**hypothèse de normalité**) et que les variances sont égales à σ, la moyenne pondérée de σ_0 et σ_1 (**hypothèse d'homoscédasticité**). Le résultat attendu est ainsi obtenu.

Egalement, repartons de l'Équation 2.10 et poursuivons en fixant $\boldsymbol{\Sigma_0} = \boldsymbol{\Sigma_1} = \boldsymbol{\Sigma}$. On obtient :

$$-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_0})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_0}) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1}) + \ln\frac{n_1}{n_0} \tag{A.8}$$

$$\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0} - \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1} \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} -\frac{1}{2}\left[\boldsymbol{\mu_1}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1} - \boldsymbol{\mu_0}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0}\right] + \ln\frac{n_1}{n_0} \tag{A.9}$$

$$\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_0}-\boldsymbol{\mu_1}) \underset{\mathcal{C}_0}{\overset{\mathcal{C}_1}{\lessgtr}} -\frac{1}{2}\left[\boldsymbol{\mu_1}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_1} - \boldsymbol{\mu_0}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_0}\right] + \ln\frac{n_1}{n_0} \tag{A.10}$$

De cette dernière équation, on retrouve les expressions du vecteur normal $\boldsymbol{w}$ (terme de gauche) et de l'ordonnée à l'origine w_o (terme de droite) dans le cadre de l'ADL, ce qui prouve l'adéquation des points de vue géométrique et probabiliste de l'analyse discriminante.

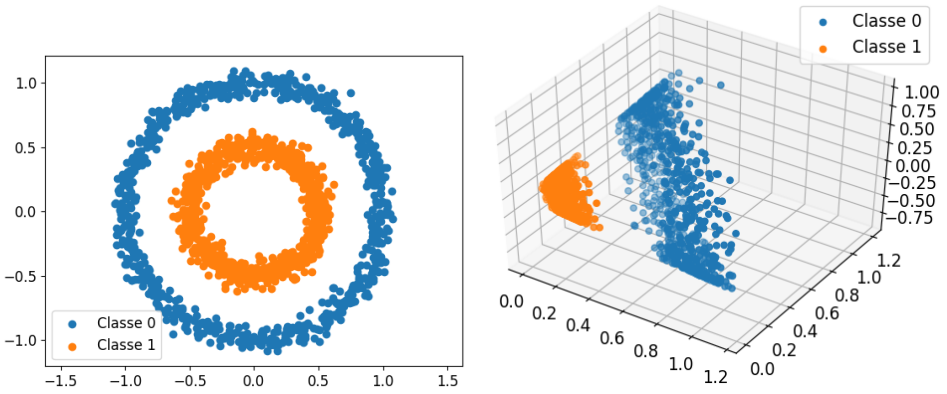
Astuce du noyau

B

Cette annexe donne davantage de détails sur l'astuce du noyau.

Cette dernière stipule que, dans le cas d'une problématique de classification non-linéaire par exemple, il est possible de déterminer un espace de représentation, en plus grande dimension que celle des données en entrée, dans lequel la tâche de classification devient linéaire. Considérons deux vecteurs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$. La fonction $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ est appelée **noyau** si on peut l'exprimer comme un produit scalaire, c'est-à-dire s'il existe une fonction $\varphi : \mathbb{R}^p \to \mathbb{R}^D$ (avec $p < D$) telle que $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle$, où $\langle \cdot, \cdot \rangle$ représente le produit scalaire. Le calcul onéreux d'un produit scalaire en grande dimension D est donc remplacé par une fonction k opérant en dimension p.

Voyons comment cela se traduit en pratique. Considérons le jeu de données bidimensionnelles affiché en Figure B.1a représentant deux cercles concentriques légèrement bruités. Dans ce problème de classification classique à deux groupes, il est impossible de déterminer une droite qui permette une séparation parfaite des différentes classes. On peut toutefois appliquer la transformation φ qui, à une donnée $\boldsymbol{x} = (x_1, x_2)^T$ de $\mathbb{R}^2$, associe le point dans $\mathbb{R}^3$ défini par $(x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$. Les données transformées sont représentées en Figure B.1b : cette fois-ci, on y distingue clairement une séparation linéaire, sous la forme d'un plan voire même d'une droite.



(a) Jeu de données original. **(b)** Jeu de données transformé.

Figure B.1. Astuce du noyau appliquée à un problème de classification bidimensionnel à deux classes.

Le produit scalaire de deux données après transformation peut s'exprimer comme suit. Considérons deux points de $\mathbb{R}^2$, $\boldsymbol{x} = (x_1, x_2)^T$ et $\boldsymbol{y} = (y_1, y_2)^T$. Ainsi :

$$\begin{aligned}
\langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T, (y_1^2, y_2^2, \sqrt{2}y_1y_2)^T \rangle && \text{(B.1)} \\
&= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 && \text{(B.2)} \\
&= (x_1y_1 + x_2y_2)^2 && \text{(B.3)} \\
&= \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2. && \text{(B.4)}
\end{aligned}$$

Cette dernière équation montre que le produit scalaire de données transformées (en grande dimension) est directement lié au produit scalaire des données originales (en faible dimension). Elle donne également l'expression de la fonction noyau :

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2. \tag{B.5}$$

Dans cet exemple simple, une formulation exacte de la fonction de transformation a été utilisée. Ce n'est toutefois pas nécessaire dans la plupart des cas. En effet, un noyau couramment employé est le noyau Gaussien, ou *Radial Basis Function (RBF) kernel* en anglais. Il est défini comme suit :

$$\forall \boldsymbol{x}, \boldsymbol{y} \quad k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}{2\sigma^2}\right), \tag{B.6}$$

où $\|.\|_2$ représente la norme Euclidienne. Considérons le cas bidimensionnel et $\sigma = 1/\sqrt{2}$ par souçi de simplicité. Alors :

$$\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{y}) &= \exp\left(-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right) && \text{(B.7)} \\
&= \exp\left(-(x_1 - y_1)^2 - (x_2 - y_2)^2\right) && \text{(B.8)} \\
&= \exp\left(-(x_1^2 + x_2^2) - (y_1^2 + y_2^2) + 2(x_1y_1 + x_2y_2)\right) && \text{(B.9)} \\
&= \exp\left(-\|\boldsymbol{x}\|_2^2\right) \exp\left(-\|\boldsymbol{y}\|_2^2\right) \exp\left(2\langle \boldsymbol{x}, \boldsymbol{y} \rangle\right). && \text{(B.10)}
\end{aligned}$$

En appliquant un développement en série entière sur cette dernière équation, on obtient :

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\|\boldsymbol{x}\|_2^2\right) \exp\left(-\|\boldsymbol{y}\|_2^2\right) \sum_{i=0}^{\infty} \frac{\left(2\langle \boldsymbol{x}, \boldsymbol{y} \rangle\right)^i}{i!}. \tag{B.11}$$

Deux remarques sont à faire sur ce résultat. Premièrement, bien que cela ne soit pas évident de prime abord, on retrouve effectivement une expression de la fonction noyau en fonction du produit scalaire des données d'origine $\boldsymbol{x}$ et $\boldsymbol{y}$. Dans un second temps, on peut observer une infinité de termes contenant ce produit scalaire (du fait du développement en série) : on en déduit que la fonction φ transforme les données

originales dans un espace en dimension **infinie**. Cela ne s'en ressent pas dans la pratique car la fonction noyau opère uniquement sur l'espace original.

Nous souhaitons enfin souligner que l'astuce du noyau est couramment employée, notamment dans les machines à vecteurs de support [CV95].

C

Expression de la matrice de covariance associée à une ellipsoïde

Dans cette annexe est apportée la démonstration du Théorème 5.1 qui précise le lien entre ellipsoïde et matrice de covariance des données situées sur le contour de cette dernière.

Avant de plonger dans les détails, nous commencons par fournir quelques résultats préliminaires utiles pour la démonnstration.

Proposition 1 (Marsaglia [Mar72])**.** *Considérons un vecteur aléatoire* $\boldsymbol{z} = (z_1, \dots, z_d)^T$ *distribué selon la loi normale multivariée* $\mathcal{N}(\boldsymbol{O_d}, \boldsymbol{I_d})$*, où* $\boldsymbol{O_d} = (0, \dots, 0)^T$ *et* $\boldsymbol{I_d}$ *est la matrice identité de* $\mathbb{R}^d$*. Alors le vecteur* $\tilde{\boldsymbol{z}} = \boldsymbol{z}/\|\boldsymbol{z}\|_2$ *est uniformément réparti à la surface de la sphère centrée sur l'origine et de rayon* 1*.*

Proposition 2 (Lehmann et Romano [LR05])**.** *Si* $X_1, \dots, X_n, Y_1, \dots, Y_m$ *sont des variables aléatoires indépendantes suivant la loi normale centrée-réduite* $\mathcal{N}(0, 1)$*, alors la variable aléatoire*

$$\frac{\left(X_1^2 + \cdots + X_n^2\right)/n}{\left(Y_1^2 + \cdots + Y_m^2\right)/m} \tag{C.1}$$

suit une loi de Fisher à n *et* m *degrés de liberté.*

Définition 1 (Fonction Beta)**.** Pour tous $x, y \in \mathbb{C}$ tels que $\mathrm{Re}(x), \mathrm{Re}(y) > 0$, la fonction Beta est définie par :

$$\mathrm{B}(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,\mathrm{d}t. \tag{C.2}$$

Elle s'exprime également, grâce à la fonction Gamma, par [Art15] :

$$\mathrm{B}(x, y) = \frac{\Gamma(x)\,\Gamma(y)}{\Gamma(x+y)}. \tag{C.3}$$

Théorème C.1. *Soit* $\mathscr{E}$ *une ellipsoïde (en dimension* d*) centrée sur* $\boldsymbol{\mu}_{\mathscr{E}}$*, alignée avec les axes canoniques de* $\mathbb{R}^d$ *et dont les demi-axes sont notés* $a_1, a_2, \dots, a_d$*. En échantillonnant*

de manière ***uniforme*** *des points au niveau de son contour, la matrice de covariance* $\boldsymbol{C}_{\mathscr{E}}$ *associée est égale à :*

$$\boldsymbol{C}_{\mathscr{E}} = \frac{1}{d}\begin{pmatrix} a_1^2 & & (0) \\ & \ddots & \\ (0) & & a_d^2 \end{pmatrix}, \tag{C.4}$$

où (0) *représente une matrice triangulaire de zéros de taille appropriée.*

Démonstration. Nous allons procéder en deux temps : d'abord, la matrice de covariance dans le cas d'une sphère de rayon 1 sera déterminée, puis nous transposerons à celui de l'ellipsoïde par transformation linéaire.

Nous nous mettons dans les conditions de la Proposition 1 et considérons un vecteur aléatoire $\boldsymbol{z} = (z_1, \dots, z_d)^T$ ainsi que celui normalisé $\tilde{\boldsymbol{z}} = \boldsymbol{z}/\|\boldsymbol{z}\|_2$. Déterminons la fonction de répartition de chaque variable du vecteur aléatoire $\tilde{\boldsymbol{z}}$. Ce qui suit se concentre sur $\tilde{z}_1$, le raisonnement est toutefois similaire pour les autres variables. De plus, par simplicité, nous nous focalisons dans un premier temps sur le cas où $-1 < x < 0$. Ainsi :

$$\mathbb{P}\left(\frac{z_1}{\sqrt{z_1^2 + \cdots + z_d^2}} \leq x\right) = \frac{1}{2}\mathbb{P}\left(\frac{z_1^2}{z_1^2 + \cdots + z_d^2} \geq x^2\right) \tag{C.5}$$

$$= \frac{1}{2}\mathbb{P}\left(z_1^2 \geq x^2(z_1^2 + \cdots + z_d^2)\right) \tag{C.6}$$

$$= \frac{1}{2}\mathbb{P}\left((1-x^2)z_1^2 \geq x^2(z_2^2 + \cdots + z_d^2)\right) \tag{C.7}$$

$$= \frac{1}{2}\mathbb{P}\left(\frac{z_1^2}{z_2^2 + \cdots + z_d^2} \geq \frac{x^2}{1-x^2}\right) \tag{C.8}$$

$$= \frac{1}{2}\mathbb{P}\left(\frac{(d-1)z_1^2}{z_2^2 + \cdots + z_d^2} \geq \frac{(d-1)x^2}{1-x^2}\right) \tag{C.9}$$

$$= \frac{1}{2}\left(1 - \mathbb{P}\left(\frac{(d-1)z_1^2}{z_2^2 + \cdots + z_d^2} \leq \frac{(d-1)x^2}{1-x^2}\right)\right) \tag{C.10}$$

La division par deux en Équation C.5 est due à la symétrie entre les cas où x est négatif et où x est positif.

Grâce à la Proposition 2, on remarque que la quantité $q = (d-1)z_1^2/(z_2^2 + \cdots + z_d^2)$ de l'Équation C.10 suit effectivement une loi de Fisher à $n = 1$ et $m = d - 1$ degrés de liberté. La probabilité intérieure s'apparente donc à la fonction de répartition de la variable aléatoire q. On peut donc poursuivre le raisonnement comme suit :

$$\mathbb{P}\left(\frac{z_1}{\sqrt{z_1^2+\cdots+z_d^2}} \leqslant x\right) = \frac{1}{2}\left(1 - I_u\left(\frac{1}{2}, \frac{d-1}{2}\right)\right) \tag{C.11}$$

$$= \frac{1}{2}\left(1 - \frac{\mathrm{B}\left(u; \frac{1}{2}, \frac{d-1}{2}\right)}{\mathrm{B}\left(\frac{1}{2}, \frac{d-1}{2}\right)}\right) \tag{C.12}$$

où $\mathrm{B}(.,.)$ est la fonction bêta, $\mathrm{B}(.;.,.)$ (respectivement I_u) est la fonction bêta incomplète (respectivement régularisée) et

$$u = \frac{1 \times (d-1)x^2/(1-x^2)}{1 \times (d-1)x^2/(1-x^2) + d - 1} = x^2. \tag{C.13}$$

Par définition, la fonction bêta incomplète de l'Équation C.12 peut s'écrire sous la forme :

$$\mathrm{B}\left(x^2; \frac{1}{2}, \frac{d-1}{2}\right) = \int_0^{x^2} t^{-1/2}(1-t)^{(d-3)/2}\,\mathrm{d}t \tag{C.14}$$

On peut ainsi calculer sa dérivée par rapport à x, grâce au théorème fondamental de l'analyse :

$$\mathrm{B}'\left(x^2; \frac{1}{2}, \frac{d-1}{2}\right) = 2x\frac{1}{\sqrt{x^2}}(1-x^2)^{(d-3)/2} \tag{C.15}$$

$$= -2(1-x^2)^{(d-3)/2} \tag{C.16}$$

Finalement, l'Équation C.12 représentant la fonction de répartition par rapport à la variable $\tilde{z}_1$, il nous suffit de dériver cette expression afin d'obtenir la densité de probabilité correspondante. Ainsi :

$$\forall x \in]-1;0[\qquad f_{\tilde{z}_1}(x) = \frac{1}{2} \times \left(-\frac{-2(1-x^2)^{(d-3)/2}}{\mathrm{B}\left(\frac{1}{2}, \frac{d-1}{2}\right)}\right) \tag{C.17}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)}\left(1-x^2\right)^{\frac{d-3}{2}} \tag{C.18}$$

Nous pouvons procéder de la même manière sur l'intervalle $]0;1[$: l'expression de la densité de probabilité est identique par symétrie. Cela rend l'Équation C.18 valable pour tout x de l'intervalle $]-1;1[$.

Maintenant que nous avons obtenu la densité de probabilité, il est aisé de déterminer son espérance ainsi que sa variance.

$$\mathbb{E}[\tilde{z}_1] = \int_{-1}^{1} x f_{\tilde{z}_1}(x) dx \tag{C.19}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \int_{-1}^{1} x \left(1 - x^2\right)^{\frac{d-3}{2}} dx \tag{C.20}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \times \frac{2}{d-1} \times \frac{1}{-2} \left[\left(1 - x^2\right)^{\frac{d-1}{2}}\right]_{-1}^{1} \tag{C.21}$$

$$= 0 \tag{C.22}$$

On trouve une espérance égale à 0 : ce résultat est rassurant car la sphère est centrée sur l'origine. Quid de la variance ?

$$\mathbb{V}[\tilde{z}_1] = \int_{-1}^{1} x^2 f_{\tilde{z}_1}(x) dx \tag{C.23}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \int_{-1}^{1} x^2 \left(1 - x^2\right)^{\frac{d-3}{2}} dx \tag{C.24}$$

Grâce à la Définition 1, nous pouvons continuer par :

$$\mathbb{V}[\tilde{z}_1] = \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} 2 \int_{0}^{1} x^2 \left(1 - x^2\right)^{\frac{d-3}{2}} dx \tag{C.25}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \int_{0}^{1} t^{1/2} \left(1 - t\right)^{\frac{d-3}{2}} dt \tag{C.26}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \frac{\Gamma(\frac{3}{2})\,\Gamma(\frac{d-3}{2}+1)}{\Gamma(\frac{3}{2}+\frac{d-3}{2}+1)} \tag{C.27}$$

$$= \frac{1}{\mathrm{B}\left(\frac{d-1}{2}, \frac{1}{2}\right)} \frac{\sqrt{\pi}\Gamma\left(\frac{d-3}{2}+1\right)}{2\Gamma\left(\frac{d-3}{2}+\frac{5}{2}\right)} \tag{C.28}$$

$$= \frac{\Gamma\left(\frac{d-1}{2}+\frac{1}{2}\right)}{\Gamma\left(\frac{d-1}{2}\right)\Gamma\left(\frac{1}{2}\right)} \times \frac{\sqrt{\pi}\Gamma\left(\frac{d-3}{2}+1\right)}{2\Gamma\left(\frac{d-3}{2}+\frac{5}{2}\right)} \tag{C.29}$$

$$= \frac{\Gamma\left(\frac{d}{2}\right)}{\Gamma\left(\frac{d-1}{2}\right)\sqrt{\pi}} \times \frac{\sqrt{\pi}\Gamma\left(\frac{d-1}{2}\right)}{2\Gamma\left(\frac{d}{2}+1\right)} \tag{C.30}$$

$$= \frac{\Gamma\left(\frac{d}{2}\right)}{2\frac{d}{2}\Gamma\left(\frac{d}{2}\right)} \tag{C.31}$$

$$= \frac{1}{d} \tag{C.32}$$

Le même raisonnement est appliqué aux autres variables de $\tilde{\boldsymbol{z}}$.

Pour résumer, nous avons démontré que, si des points sont échantillonnés de manière uniforme à la surface de la sphère centrée sur l'origine et de rayon 1, alors leur matrice de covariance est égale à $\frac{1}{d}\boldsymbol{I_d}$, où $\boldsymbol{I_d}$ est la matrice identité en dimension d.

Afin de passer de la sphère à l'ellipsoïde, nous appliquons une transformation linéaire aux données situées sur la sphère unitaire, comme suggéré par Dezert et Musso [DM01]. Cette transformation est paramétrée par les demi-axes $a_1, \ldots, a_d$ de l'ellipsoïde. Ainsi, chaque variable est multipliée par le demi-axe correspondant. Or on sait que, si X est une variable aléatoire et a un scalaire, alors $\mathbb{V}[bX] = b^2\mathbb{V}[X]$. On obtient donc une variance égale à $a_i^2/d \quad \forall i \in \{1, \ldots, d\}$. Finalement, cela permet d'écrire la matrice de covariance des points situés sur le contour de l'ellipsoïde $\mathscr{E}$ sous la forme :

$$\boldsymbol{C}_{\mathscr{E}} = \frac{1}{d}\begin{pmatrix} a_1^2 & & (0) \\ & \ddots & \\ (0) & & a_d^2 \end{pmatrix} \tag{C.33}$$

□

Annotations manuelles

D

Nous détaillons dans cette annexe l'effort d'annotation réalisé durant cette thèse.

En effet, les vérités-terrains utilisées jusqu'ici par le projet SIVAO consiste en des annotations de vidéos, passage par passage : c'est-à-dire indice, nombre de passagers, position (haute ou basse) du garde-corps, *etc*. Elles ont été générées manuellement, au fil du temps, par les employés de BLUECIME. Ces annotations sont utilisées en pratique pour valider ou non une nouvelle version de l'algorithme, vis-à-vis des résultats finaux obtenus. Toutefois, elles sont réalisées uniquement sur les images unitaires : en d'autres mots, la première partie du système SIVAO (*i.e.* la détection et le suivi des véhicules dans la vidéo) n'est donc jamais évaluée.

C'est pourquoi nous avons décidé de travailler sur la création d'une base de données constituées d'images issues directement de la caméra pour lesquelles chaque véhicule et chaque passager sont annotés. Nous nous sommes donc associés à une entreprise spécialisée dans ce domaine, à savoir ISAHIT [1]. Cette dernière a pour but de mettre en relation des sociétés françaises et des travailleurs de pays d'Afrique afin d'externaliser certaines tâches numériques.

Après de multiples échanges, le protocole final a été fixé comme suit :

— Les images concernées sont celles issues de la caméra.

— Deux types d'annotations sont considérées : celles des véhicules ainsi que celles des passagers de ces derniers. Ceci exclut donc tout autre skieur présent sur les pistes ou en attente d'embarquement sur la remontée mécanique.

— Les détourages sont effectués sur les contours extérieurs, *i.e.* les "trous" (pour l'intérieur des véhicules notamment) ne sont pas pris en compte.

— Chaque véhicule est indexé par rapport à sa position dans l'image vis-à-vis du câble principal de la remontée mécanique, en commencant à 1. En l'occurence, le sens choisi est horaire, c'est-à-dire que le câble est suivi de la gauche de l'image jusqu'au pylône central, puis jusqu'à la sortie du véhicule de l'image à droite. Une illustration est donnée en Figure D.1.

— Pour chaque véhicule, le nombre de passagers présents est noté.

1. https://isahit.com/

— En ce qui concerne les personnes, trois informations nous intéressent : le “type” du passager (adulte ou enfant), sa position relative sur le siège et l’indice du véhicule sur lequel il est assis (comme décrit ci-dessus).

— Pour ce qui est de la position relative, elle est évaluée de gauche à droite par rapport au siège. Cela veut dire que le passager le plus à gauche est indexé 1, celui à droite 2, *etc*.

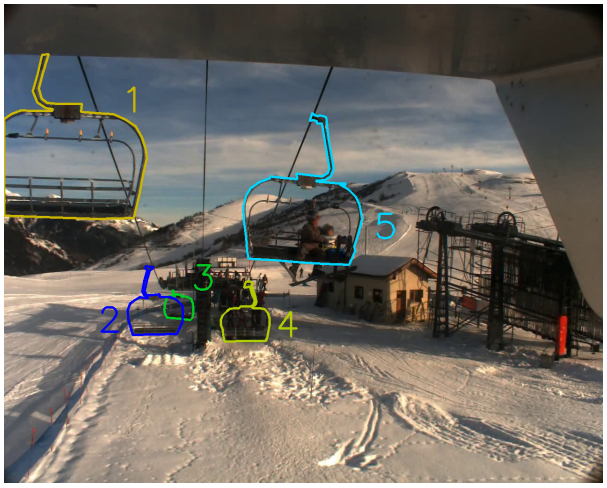


Figure D.1. Ordre d’indexation des véhicules. Chaque siège est détouré d’une couleur, l’indice associé est affiché de la même couleur à côté.

Quantitativement, nous avons choisi de faire annoter 38 vidéos, d’une durée de deux minutes, issues de 21 remontées mécaniques différentes. Elles sont représentatives de tout genre de situations, notamment en termes de conditions météorologiques.

Deux types d’images sont extraites d’une même vidéo :

- Au plus 100 images échantillonnées de manière égale tout au long de la vidéo.
- Deux séquences de 40 images **consécutives** dont le début est choisi de manière aléatoire au sein de la vidéo.

Au final, 6000 images sont obtenues, également réparties entre les deux catégories ci-dessus.

Les fichiers retournés par ISAHIT sont au format JavaScript Object Notation (JSON). Ils comprennent différentes informations utiles, notamment le contour des objets détourés sous forme de polygônes, c’est-à-dire une suite de coordonnées de pixels. Il s’agit en réalité d’une liste de polygônes, éventuellement à un seul élément, afin de prendre en compte le fait que certains véhicules peuvent être décomposés en plusieurs sous-parties (par exemple lorsqu’ils passent derrière un pylône).

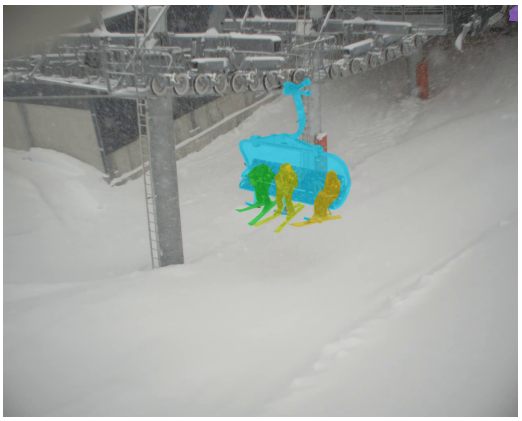
Nous donnons en Figure D.2 un extrait de fichier JSON retourné par ISAHIT pour une seule image. On retrouve en ligne 7 les différents objets annotés sous forme d'une liste. Les caractéristiques du premier objet sont donnés en ligne 11, 12 et 13 : on a ici affaire à un véhicule, dont l'indice est 2 et qui transporte trois passagers. Les lignes 19 à 21 et 27 à 29 précisent les contours de l'objet en question, sous la forme de polygônes. La boîte englobante est également fournie en ligne 34. On peut finalement noter que quatre autres éléments sont annotés (lignes 41 à 44).

Quatre exemples d'images retournées par ISAHIT sont donnés en Figure D.3. Chaque couleur représente un objet différent.

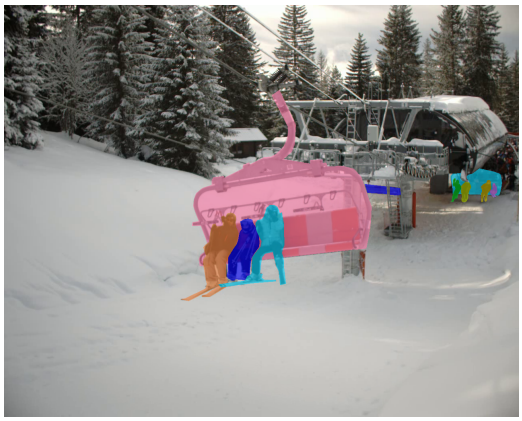
A l'heure actuelle, ces annotations restent en utilisation interne, au sein même du projet MIVAO. Nous espérons pouvoir prochainement les mettre à disposition de la communauté scientifique, cette base de données présentant en effet nombre de difficultés intéressantes.

```
{
    "importDate": "2020-03-06 17:38",
    "id": 4682607,
    "user": "aandrianiaina",
    "customerRef": "image_5432.png",
    "timeSpent": 74,
    "annotations": [
        {
            "area": 42548.5,
            "labels": {
                "object_type": "vehicle",
                "index": "vehicle_2",
                "npassengers": "3 passengers"
            },
            "color": "255, 213, 0",
            "polygons": [
                {
                    "points": [
                        [758, 179],
                        ...,
                        [753, 183]
                    ],
                    "type": "polygon"
                },
                {
                    "points": [
                        [748, 526],
                        ...,
                        [747, 532]
                    ],
                    "type": "polygon"
                }
            ],
            "boundingbox": [
                [586, 179],
                [864, 179],
                [864, 536],
                [586, 536]
            ]
        },
        {...},
        {...},
        {...},
        {...}
    ],
    "dateDone": "2020-04-04 06:27",
    "taskStatus": 1
}
```

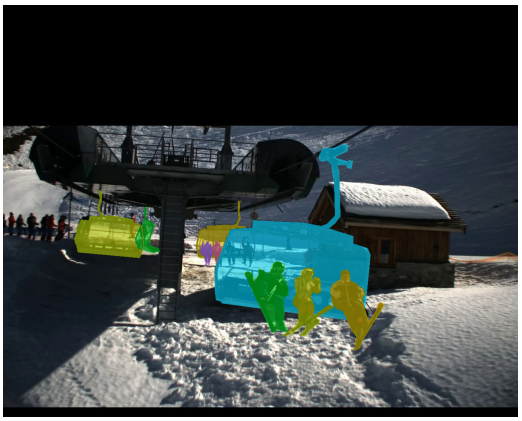

Figure D.2. Extrait de fichier JSON retourné.



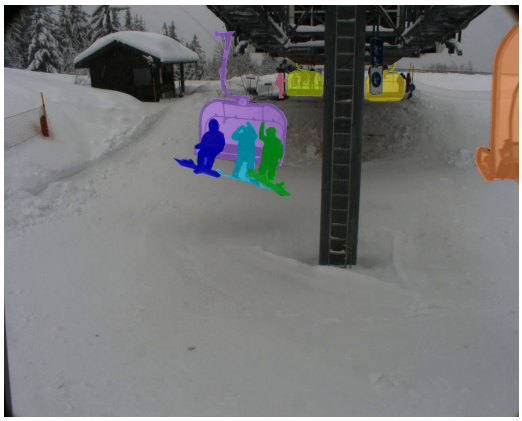
(a) Exemple 1.



(b) Exemple 2.



(c) Exemple 3.



(d) Exemple 4.

Figure D.3. Exemples d'images annotées manuellement.