Calage robuste et accéléré de nuages de points en environnements naturels via l'apprentissage automatique

Mémoire

Maxime Latulippe

Maîtrise en informatique
Maître ès sciences (M.Sc.)

Québec, Canada

Résumé

En robotique mobile, un élément crucial dans la réalisation de la navigation autonome est la localisation du robot. En utilisant des scanners laser, ceci peut être réalisé en calant les nuages de points consécutifs. Pour ce faire, l'utilisation de points de repères appelés descripteurs sont généralement efficaces, car ils permettent d'établir des correspondances entre les nuages de points. Cependant, nous démontrons que dans certains environnements naturels, une proportion importante d'entre eux peut ne pas être fiable, dégradant ainsi les performances de l'alignement. Par conséquent, nous proposons de filtrer les descripteurs au préalable afin d'éliminer les nuisibles. Notre approche consiste à utiliser un algorithme d'apprentissage rapide, entraîné à la volée sous le paradigme *positive and unlabeled learning* sans aucune intervention humaine nécessaire. Les résultats obtenus montrent que notre approche permet de réduire significativement le nombre de descripteurs utilisés tout en augmentant la proportion de descripteurs fiables, accélérant et augmentant ainsi la robustesse de l'alignement.

Abstract

Localization of a mobile robot is crucial for autonomous navigation. Using laser scanners, this can be facilitated by the pairwise alignment of consecutive scans. For this purpose, landmarks called descriptors are generally effective as they facilitate point matching. However, we show that in some natural environments, many of them are likely to be unreliable. The presence of these unreliable descriptors adversely affects the performances of the alignment process. Therefore, we propose to filter unreliable descriptors as a prior step to alignment. Our approach uses a fast machine learning algorithm, trained on-the-fly under the *positive and unlabeled* learning paradigm without the need for human intervention. Our results show that the number of descriptors can be significantly reduced, while increasing the proportion of reliable ones, thus speeding up and improving the robustness of the scan alignment process.

Table des matières

Liste des tableaux

Liste des figures

Avant-propos

La tradition veut qu'un mémoire débute presque inévitablement par une kyrielle de remerciements. Ces deux années de labeur étant presque achevées, il m'apparaît finalement tout à fait normal de vouloir souligner l'apport, direct ou indirect, de certaines personnes à l'accomplissement de ma maîtrise.

Il me faut d'abord remercier particulièrement mon superviseur de recherche, Philippe Giguère, ayant fait preuve d'une grande confiance et disponibilité à mon égard. Les nombreuses opportunités qu'il m'a offertes (y compris celle d'aller en Chine !) auront sans aucun doute été très pertinentes pour compléter ma formation. Je tiens par ailleurs à souligner ses nombreux conseils, autant techniques que de vie, qui ont su me faire réfléchir et m'orienter.

Je dois également remercier Alexandre Drouin et le professeur François Laviolette, membres du Groupe de Recherche en Apprentissage Automatique de l'Université Laval (GRAAL), pour leur aide considérable dans l'élaboration de la solution au problème abordé dans ce mémoire. Je remercie de plus certains autres membres du GRAAL, en particulier Sébastien Giguère, Alexandre Lacoste, Pascal Germain et Jean-Francis Roy, pour toutes ces discussions générant des idées et en confrontant d'autres, mais aussi (il faut l'avouer) ces inéluctables moments de procrastination à parler de tout autres sujets. À cela, il ne faudrait surtout pas oublier d'ajouter le membre non officiel mais tout aussi important qu'est la machine à café du GRAAL qui, presque quotidiennement, a su m'apporter éveil et réconfort !

Finalement, je remercie ma copine Jany pour sa grande compréhension et ses encouragements pendant mes périodes de travail acharné où ma disponibilité tendait asymptotiquement vers zéro.

Chapitre 1

Introduction

Le domaine de la robotique mobile gagne en importance depuis plusieurs années. Le développement de nouvelles technologies et d'algorithmes s'effectue à un rythme élevé, permettant ainsi aux robots d'accomplir des tâches de plus en plus complexes. Un défi important dans ce domaine consiste à rendre les robots capables d'opérer et de s'adapter de façon autonome à leur environnement. Ceci pourrait éventuellement permettre de remplacer les humains pour effectuer des tâches dangereuses, ou qui doivent être réalisées dans un environnement hostile ou inaccessible pour l'humain.

Un élément fondamental pour l'atteinte de cet objectif est la navigation autonome d'un robot dans un environnement inconnu. Pour accomplir une telle tâche, la localisation du robot est un aspect crucial. Cette localisation doit être suffisamment précise pour permettre, entre autres, de cartographier correctement l'environnement et de suivre des itinéraires vers des destinations d'intérêt.

Alors qu'une solution simple consiste à utiliser un système de localisation globale, tel qu'un *Global Positioning System* (GPS) ou un *Differential GPS* (DGPS), leur utilisation est parfois limitée ou impossible (forêts, mines souterraines, autres planètes, etc.). Par conséquent, une autre solution consiste à mesurer les déplacements effectués par le robot afin de suivre sa trajectoire et, par le fait même, de le localiser dans l'environnement. Pour ce faire, des capteurs tels qu'odomètre, accéléromètre ou gyroscope peuvent être utilisés pour mesurer ces déplacements, en translation et en rotation. Ces capteurs, dits *proprioceptifs* car ils acquièrent de l'information sur l'état du robot lui-même et non sur son environnement extérieur, sont néanmoins sujets à une dérive des mesures. En effet, la variance des propriétés des composantes électroniques et mécaniques utilisées pour leur fabrication engendre un biais dans les mesures. Ce biais est généralement quantifié par calibration, mais varie selon les conditions d'opération des capteurs (température, voltage d'alimentation, vibrations, dérapage des roues, etc.). Le décalage des mesures est donc inévitable et rend les capteurs proprioceptifs imprécis pour suivre la position d'un robot à long terme.

Une localisation plus précise à long terme peut se réaliser en utilisant des capteurs *extéroceptifs*, tel que les caméras, car ils acquièrent des informations sur l'environnement extérieur au robot. Les objets ou régions particulières qui sont observés dans l'environnement peuvent alors servir de points de repère, par rapport auxquels le robot peut se localiser. Ceci permet conséquemment de quantifier plus précisément les déplacements du robot. Dans ce mémoire, nous nous intéressons à la localisation effectuée avec un scanner laser, capteur qui a l'avantage de percevoir en trois dimensions l'environnement de manière relativement précise. Ce type de capteurs est d'ailleurs récent et très populaire.

Les scanners laser permettent d'acquérir des ensembles de mesures de distances, par rapport au robot, des objets et régions environnantes. Un ensemble de ces mesures, pouvant chacune être localisée dans l'espace 3D dans le repère local du capteur, permet d'obtenir un *nuage de points*. Les nuages de points représentent donc un échantillonnage plus ou moins précis de l'environnement entourant le robot, dépendant du modèle de scanner laser.

Pour localiser un robot en utilisant un scanner laser, les nuages de points consécutifs recouvrant partiellement une même partie de l'environnement peuvent être alignés grâce à leur zone de recouvrement. Cette opération d'alignement, appelée le *calage*, permet d'estimer les déplacements effectués par le robot entre les nuages de points. Ces déplacements peuvent être décomposés par une translation $\mathbf{t}$ et une rotation $\mathbf{R}$. Par conséquent, le problème de calage peut être défini comme étant la recherche des valeurs de $\mathbf{t}$ et $\mathbf{R}$ alignant le mieux les nuages de points. Alors que ce problème est bien maîtrisé dans les environnements structurés (corridors, bâtiments, etc.), il reste difficile dans les environnements naturels non structurés comme les forêts.

Pour une raison qui sera détaillée plus loin dans ce mémoire, le calage s'effectue la plupart du temps en deux étapes : un alignement approximatif initial, appelé *calage grossier*, suivi d'un ajustement plus précis de l'alignement, appelé *calage fin*. Alors que le calage fin est souvent réalisé avec le même algorithme (*Iterative Closest Point* présenté plus loin) ou l'une de ses variantes, il existe de nombreuses méthodes pour réaliser le calage grossier, qui permettent d'adapter le problème à différents types d'environnements, d'objets ou de capteurs.

L'une de ces méthodes consiste à calculer, dans les nuages de points, des ensembles de valeurs caractéristiques décrivant la géométrie locale autour de chacun des points. Ces ensembles de valeurs, appelés *descripteurs*, peuvent ensuite être comparés entre les nuages de points et pairés lorsqu'ils sont semblables. Ces pairages permettent d'établir des correspondances entre les nuages de points, dans l'espoir que les points des descripteurs pairés soient situés au même endroit dans l'environnement. Les correspondances établies entre deux nuages de points peuvent alors être utilisées pour calculer un alignement approximatif, en trouvant les valeurs de $\mathbf{R}$ et $\mathbf{t}$ qui superposent les points correspondants.

Le problème abordé dans ce mémoire est que, dans les environnements naturels, les descrip-

teurs de points situés dans le feuillage sont en grande majorité non fiables pour établir des correspondances. La présence de tels descripteurs affecte donc négativement les performances du calage en termes de robustesse, de précision et de temps de calcul.

Pour améliorer les performances du calage dans ce type d'environnement, nous proposons une méthode pour filtrer les descripteurs après leur calcul. Cette méthode, basée sur l'apprentissage automatique, consiste à entraîner un classificateur afin d'être en mesure de distinguer, pour un environnement donné, les descripteurs fiables de ceux qui ne le sont pas. L'objectif est d'éliminer le plus possible de descripteurs non fiables tout en conservant le plus possible ceux qui sont fiables. L'entraînement est effectué d'après un paradigme d'apprentissage appelé *positive and unlabeled learning*, permettant d'entraîner un classificateur à partir d'un ensemble de données d'entraînement contenant uniquement des exemples positifs et des exemples non étiquetés. Ce paradigme a été choisi pour des raisons qui seront évoquées plus loin. Un aspect essentiel de la méthode est que les données d'entraînement sont générées et étiquetées par le robot lui-même, sans aucune intervention humaine. Ceci permet au robot de s'entraîner à la volée dans un nouvel environnement. Cet avantage pourrait permettre au robot de s'adapter naturellement aux changements de l'environnement à mesure qu'il se déplace, soit en se réentraînant périodiquement, soit lorsque jugé nécessaire.

La méthode de filtrage proposée a été testée sur deux ensembles de données acquis dans des environnements extérieurs, dont l'un présentant une végétation dense. Les résultats obtenus montrent une diminution du nombre d'erreurs importantes d'alignement, se traduisant ainsi par une amélioration de la robustesse du calage, de même qu'une réduction significative du temps de calcul requis.

Par ailleurs, la méthode que nous proposons a fait l'objet d'une publication dans *23rd International Joint Conference on Artificial Intelligence* (IJCAI 2013) [22]. À notre connaissance, l'apprentissage automatique n'a pas été utilisé auparavant pour filtrer les descripteurs calculés dans des nuages de points afin de ne conserver que les plus fiables, dans un certain type d'environnement, pour améliorer les performances de l'estimation de correspondances et du calage.

La suite de ce mémoire est organisée comme suit :

- Le chapitre 2 introduit d'abord plus formellement la notion de nuages de points, de même que certains traitements de base fréquemment effectués sur ces derniers. Le problème de calage est ensuite détaillé et quelques algorithmes permettant de le résoudre sont présentés. Enfin, les concepts de descripteurs et d'estimation de correspondances sont définis, suivis de la présentation de quelques principaux types de descripteurs proposés dans la littérature.
- Le chapitre 3 aborde quelques notions de base de l'apprentissage automatique dont la compréhension sera nécessaire lors de la présentation de la méthode de filtrage proposée. Quelques algorithmes d'apprentissage connus y sont également présentés.

– Le chapitre 4 présente en premier lieu un aperçu de la méthode proposée, ainsi qu'une revue de littérature des travaux connexes publiés précédemment. En deuxième lieu, le problème de la présence de descripteurs non fiables en environnements naturels est expliqué et démontré en détails. Une méthode existante de filtrage des descripteurs, qui est toutefois peu adaptée aux environnements naturels, est ensuite présentée, suivie de la description de la méthode de filtrage que nous proposons.
– Le chapitre 5 présente la structure des tests réalisés pour évaluer la méthode de filtrage proposée, de même que les résultats obtenus.
– Le chapitre 6 conclut le mémoire et énonce quelques avenues possibles pour les travaux futurs.

Chapitre 2

Notions de calage

Ce chapitre détaille d'abord la notion de nuages de points, de même que certains traitements de base qui leur sont propres et qui sont fréquemment utilisés pour des opérations plus complexes comme le calage. Le problème de calage est ensuite décrit plus formellement suivi de quelques méthodes pour le résoudre. Enfin, la dernière section porte sur la notion de descripteurs, soit l'extraction de caractéristiques dans les nuages de points, qui permettent d'établir des correspondances entre des nuages de points et facilitent ainsi le calage.

2.1 Nuages de points

Un nuage de points (*scan*) est une structure de données servant à représenter un ensemble de points multidimensionnels. Dans le contexte de scanners laser 3D, chaque point possède une coordonnée en x, y et z. Certains capteurs permettent d'obtenir de l'information supplémentaire sur les points mesurés, comme par exemple la valeur d'intensité de retour du laser ou la couleur de la surface. Dans ce cas, le nuage de points peut être en quatre dimensions, voire plus. De manière générale, un nuage de points 3D peut être représenté par :

$$\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\} \tag{2.1}$$

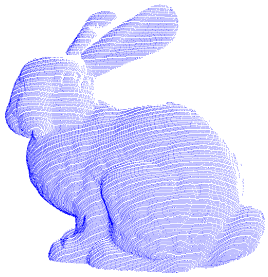
où un point p_i est décrit par ses coordonnées :

$$\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T \tag{2.2}$$

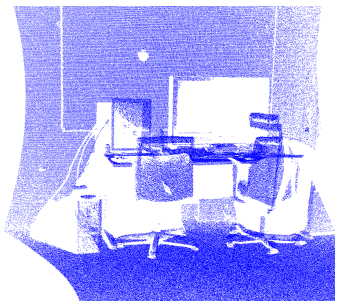
Dans le cadre de ce mémoire, les nuages de points utilisés seront limités à trois dimensions. La Figure 2.1 présente des exemples de nuages de points 3D pour : a) modèle de lapin, b) scène intérieure (bureau) et c) scène extérieure.

2.1.1 Scanners laser

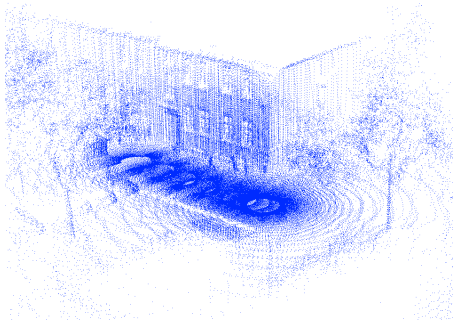
Le principe de base du fonctionnement d'un scanner laser est de faire pivoter un (ou plusieurs) faisceau laser et de prendre des mesures de distance à intervalles réguliers. À chaque mesure



(a) Modèle de lapin *Stanford Bunny* [8].



(b) Scène intérieure : bureau, tiré de *Kitchen Dataset* [35]



(c) Scène extérieure : bâtiment et arbres, tiré de *Hannover Dataset* [50]

FIGURE 2.1: Exemples de nuages de points.

de distance r correspond donc une orientation du laser θ (horizontalement) et ϕ (verticalement). Les données brutes acquises avec le capteur sont donc en coordonnées sphériques (θ, ϕ, r), mais la plupart du temps immédiatement converties en coordonnées cartésiennes (x, y, z), d'où la représentation d'un nuage de points suggérée ci-haut. Étant donné ce mode de fonctionnement, la densité des points mesurés avec ce type de capteurs diminue en fonction de leur distance par rapport au capteur. Également, la largeur de l'intervalle entre les mesures définit la résolution angulaire du scanner et, par conséquent, la précision de son échantillonnage des surfaces dans l'environnement.

Des modèles de scanners laser sont présentés à la Figure 2.2 à titre d'exemples. Les modèles présentés en (a) [1] et (b) [2] sont des scanners 2D (valeur de ϕ constante), alors que le modèle présenté en (c) [3] est un scanner 3D. Le Tableau 2.1 détaille certaines caractéristiques de ces capteurs. En particulier, le champ de vision et la portée influencent grandement le type d'applications d'un scanner laser. La résolution est également à considérer, puisqu'elle détermine la quantité d'information mesurée sur chaque objet de l'environnement, et n'est pas nécessairement la même en θ et en ϕ. Il est important de mentionner que, peu importe le modèle utilisé, il existe toujours une incertitude (bruit) sur les mesures. Ce bruit de mesure se caractérise en angle et en distance : pour chaque mesure, il y a une incertitude sur l'orientation du laser de même que sur la distance mesurée. Les algorithmes de traitement de nuages de points devraient par conséquent être développés de manière à être robustes au bruit. En d'autres mots, la sortie d'un algorithme ne devrait pas changer en présence du bruit. Enfin, le nombre de points mesurés par seconde varie grandement d'un modèle à l'autre. Un nombre plus élevé signifie plus d'information acquise sur l'environnement, mais implique en contrepartie un traitement plus lourd.

1. Hokuyo URG-04LX : `http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html`
2. Site web de Sick : `http://www.sick.com/`
3. Velodyne HDL-64E : `http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx`

(a) Hokuyo URG-04LX. (b) Sick LMS 511. (c) Velodyne HDL-64E.

FIGURE 2.2: Exemples de modèles de scanners laser.

Modèle	Champ de vision (°)	Portée (m)	Résolution angulaire (°)	Erreur de mesure (σ) (mm)	points/s
Hokuyo	240	5.6	0.352	30	∼6830
Sick	190	80	0.167 à 1	25 à 50	19000
Velodyne	360×26.8	120	0.09×0.4	20	$\sim 1.3\times10^6$

TABLE 2.1: Caractéristiques principales de quelques modèles de scanners laser.

Par ailleurs, il existe d'autres types de capteurs, tels que les caméras temps de vol [4] et la caméra Kinect de Microsoft [5], qui permettent d'acquérir des *images de profondeur*. Ces dernières peuvent être vues comme des nuages de points organisés en lignes et en colonnes, formant une matrice où chaque « pixel » possède une valeur de distance mesurée par rapport au capteur. Il ne sera pas question de ce type de capteur dans cet ouvrage.

2.2 Traitements de base

Cette section présente des traitements de base, propres aux nuages de points, qui seront utilisés plus loin pour résoudre le problème de calage.

2.2.1 Plus proche voisin

Pour un espace de données quelconque à D dimensions, le plus proche voisin d'un point $\mathbf{p}_i$ est le point $\mathbf{q}_j$ dont la distance $d_{i,j} = \|\mathbf{p}_i - \mathbf{q}_j\|$ est la plus petite, selon une métrique de distance donnée. Les résultats obtenus varient en fonction de la métrique de distance utilisée. À titre informatif, la forme générale pour le calcul d'une distance entre deux points à D dimensions $\mathbf{p} = (p_1, p_2, ..., p_D)$ et $\mathbf{q} = (q_1, q_2, ..., q_D)$, connue sous le nom de *distance de Minkowski*, est

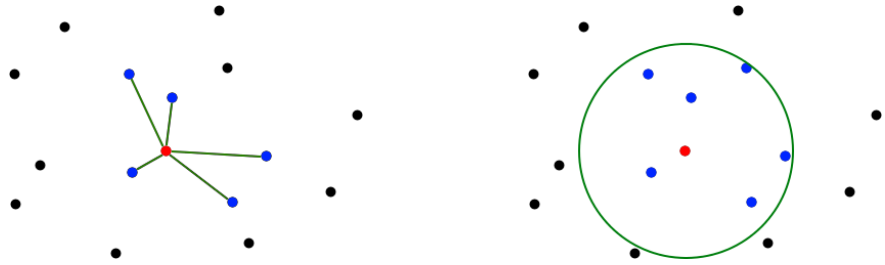
4. Voir `http://en.wikipedia.org/wiki/Time-of-flight_camera`
5. Voir `http://en.wikipedia.org/wiki/Kinect`

donnée par

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| = \left(\sum_{k=1}^{D} |p_k - q_k|^p \right)^{1/p} \tag{2.3}$$

où le paramètre p permet de contrôler l'influence des dimensions pour lesquelles la magnitude est la plus grande. Des cas particuliers couramment utilisés sont la distance de Manhattan ($p = 1$), la distance euclidienne ($p = 2$) et la distance de Tchebychev ($p \to \infty$).

Il arrive fréquemment qu'une étape de calcul d'un algorithme requiert les k plus proches voisins (k-PPV) d'un point $\mathbf{p}$. On dit alors que l'ensemble de ces points forme le *voisinage* de $\mathbf{p}$. Une autre façon de définir le voisinage d'un point est d'identifier tous les points situés à une distance inférieure à un rayon donné. Cette distinction est illustrée à la Figure 2.3. La définition du voisinage par les k-PPV a l'avantage d'être auto-adaptative à l'échelle et à la densité des points. La définition par un rayon de voisinage est quant à elle sensible à ces variations, mais procure une signification physique aux voisins trouvés.



(a) 5-PPV. (b) Rayon de calcul.

FIGURE 2.3: Illustration du calcul du voisinage d'un point (en rouge) selon a) les 5-PPV et b) un rayon donné. Les points voisins sont les points en bleu.

Le calcul des plus proches voisins est habituellement réalisé en utilisant un *k-d tree*[6] ou un *octree*[7], structures de données permettant d'organiser les points dans l'espace. Sans donner plus de détails sur le fonctionnement de ces structures, cette solution permet de déterminer rapidement un plus proche voisin, sans devoir tester tous les points. Une fois ces structures générées pour un ensemble de points, la recherche d'un plus proche voisin peut s'effectuer en $O(\log n)$, comparativement à $O(n^2)$ si tous les points sont testés.

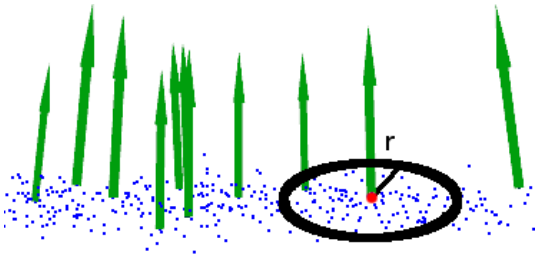
2.2.2 Estimation des normales de surface

Pour un point $\mathbf{p}_i$ situé sur une surface $\mathbf{\Phi}$ donnée, la normale $\mathbf{n}_i$ est le vecteur perpendiculaire à la surface à ce point. Dans un nuage de points, les points mesurés représentent un échantillonnage d'une surface sous-jacente inconnue. Les normales exactes de cette surface

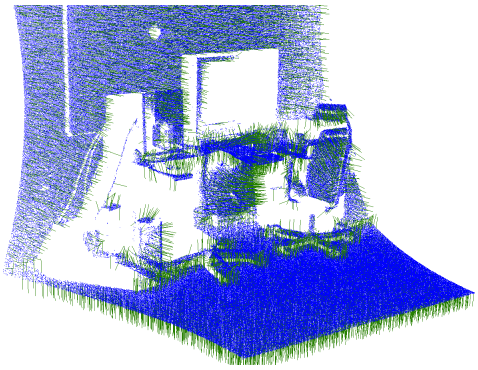
6. *k-dimensional tree*, voir `http://en.wikipedia.org/wiki/K-d_tree`.
7. Voir `http://en.wikipedia.org/wiki/Octree`.

sont donc elles aussi inconnues, mais peuvent être estimées à chacun des points en fonction de leur voisinage. Par exemple, la normale d'un point $\mathbf{p}_i$ peut être vue comme le produit croisé entre les vecteurs menant $\mathbf{p}_i$ à chacun de ses deux plus proches voisins. Cette approche étant cependant très sensible au bruit de mesure, il est préférable d'utiliser un voisinage plus grand. Comme les normales ont une signification physique, il est généralement adéquat de définir ce voisinage par un rayon. La Figure 2.4(a) illustre l'estimation des normales à partir du voisinage d'un point.

Il existe plusieurs méthodes permettant d'estimer les normales. L'une des approches les plus simples est d'effectuer une décomposition des valeurs propres et vecteurs propres [8] des coordonnées des points du voisinage de $\mathbf{p}_i$. Le vecteur propre correspondant à la valeur propre la plus petite (direction dans laquelle la variation des coordonnées est la plus faible) approxime alors $\mathbf{n}_i$. Il existe évidemment des approches plus complexes et plus précises. L'article [20] présente une synthèse des principales méthodes publiées dans la littérature et compare leurs performances. La Figure 2.4(b) présente un exemple de résultat d'estimation des normales pour la scène de bureau présentée précédemment à la Figure 2.1(b).



(a) Illustration de l'estimation des normales. Image tirée et adaptée de [37].



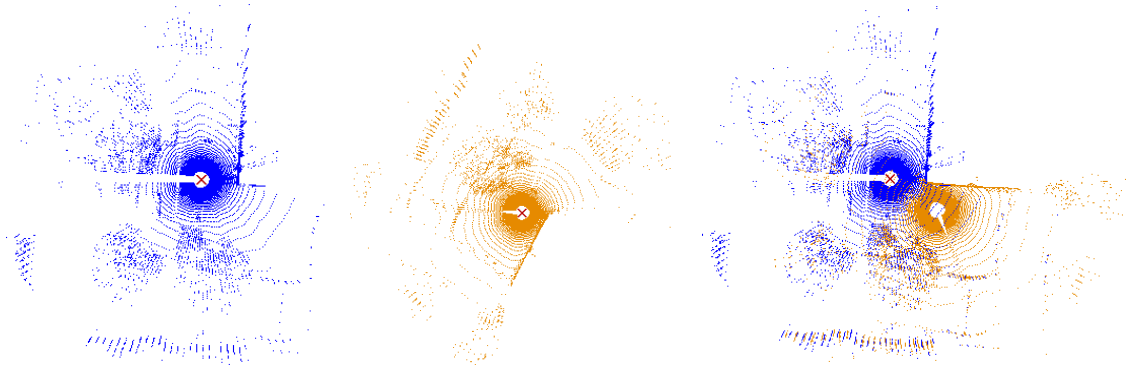
(b) Exemple de résultat d'estimation des normales pour une scène de bureau, scan original tiré de *Kitchen Dataset* [35].

FIGURE 2.4: Illustration de l'estimation des normales et exemple de résultat. Les normales sont représentées par les flèches vertes.

2.3 Calage de nuages de points

Pour un robot mobile qui se déplace, les nuages de points consécutifs acquis avec un scanner laser capturent successivement en partie une même région de l'environnement, mais de points de vue différents. Il existe donc un recoupement, plus ou moins important, entre les nuages de points. Le calage est le problème d'alignement des nuages de points sur leur zone de recoupement, afin de combiner l'information contenue dans chacun de façon cohérente. Un

8. Voir `http://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix` pour plus de détails.



(a) Premier scan. (b) Deuxième scan. (c) Calage réussi.

FIGURE 2.5: Exemple de calage réussi entre deux nuages de points (vus de haut) dans une scène extérieure. a) et b) sont les deux scans capturés par le robot. c) présente le résultat d'un bon calage entre les deux.

exemple de calage réussi entre deux nuages de points est présenté à la Figure 2.5. Au départ, chacun des nuages de points est représenté dans le repère local du robot, dont l'origine est marquée par un X rouge. Dans cet exemple, pour réaliser le calage, le nuage de points en jaune a subi une rotation et une translation afin d'être aligné sur le nuage de points bleu dans le repère local de ce dernier. Résoudre le problème de calage revient donc à déterminer la bonne transformation géométrique à appliquer aux points d'un des nuages. Pour des nuages de points acquis par un robot mobile se déplaçant dans un environnement, cette transformation géométrique est toujours rigide [9] et peut par conséquent être définie uniquement par une rotation $\mathbf{R}$ et une translation $\mathbf{t}$. Si le nuage de points à transformer, $\mathcal{P}_0$, est représenté sous forme de matrice (chaque colonne étant constituée des coordonnées d'un point), alors la transformation géométrique peut être effectuée selon l'équation suivante :

$$\mathcal{P}_{calé} = \mathbf{R}\mathcal{P}_0 + \mathbf{t} \tag{2.4}$$

où $\mathbf{R}$ est la matrice de rotation et $\mathbf{t}$ le vecteur de translation.

En déterminant la rotation et la translation entre deux nuages de points, l'opération de calage permet de déduire le déplacement effectué par le robot entre ces nuages de points et facilite, entre autres, la création d'une carte de l'environnement. Il s'agit donc d'une étape cruciale dans la réalisation de la navigation autonome à l'aide d'un scanner laser. Une autre utilité du calage, qui ne sera pas développée ici, est la localisation d'un robot dans un environnement, en utilisant une carte déjà existante. Dans ce cas, un nuage de point acquis par le robot est calé avec un modèle de l'environnement. La région sur laquelle le nuage de points est calé correspond à la position du robot.

9. Il n'y a pas de déformation ou de facteur d'échelle entre les nuages de points.

Le problème de calage se traduit habituellement par un problème d'optimisation. Pour des valeurs de $\mathbf{R}$ et $\mathbf{t}$ données, on peut évaluer la qualité du calage résultant en utilisant une fonction d'erreur $\varepsilon(\mathbf{R}, \mathbf{t})$. Cette fonction d'erreur retourne idéalement une valeur faible lorsque le calage est bon et une valeur croissante à mesure que le calage se détériore. On peut supposer que, sauf en présence de symétrie dans l'environnement (comme un corridor), le minimum global de la fonction d'erreur correspond à la solution réelle. Par conséquent, le problème consiste à trouver $\mathbf{R}$ et $\mathbf{t}$ tel que

$$\underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}}\, \varepsilon(\mathbf{R}, \mathbf{t}) \tag{2.5}$$

D'après [25], on peut d'ailleurs voir le problème général du calage comme étant la recherche de la transformation géométrique $\mathbf{R}$ et $\mathbf{t}$ minimisant l'erreur donnée par

$$\varepsilon(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{|\mathcal{P}|} d^2 \left(\mathbf{R}\mathbf{p}_i + \mathbf{t}, \mathbf{\Phi}_{\mathcal{Q}}\right) \tag{2.6}$$

où $d^2 \left(\mathbf{R}\mathbf{p}_i + \mathbf{t}, \mathbf{\Phi}_{\mathcal{Q}}\right)$ est la distance entre un point $\mathbf{p}_i$ transformé du nuage de points $\mathcal{P}$ et la surface $\mathbf{\Phi}_{\mathcal{Q}}$ sous-jacente aux points du nuage $\mathcal{Q}$. La surface $\mathbf{\Phi}_{\mathcal{Q}}$ réelle étant inconnue, la fonction de distance d^2 doit être approximée. L'une des heuristiques les plus simples pour ce faire consiste par exemple à calculer la distance euclidienne entre le point transformé $(\mathbf{R}\mathbf{p}_i + \mathbf{t})$ et son plus proche voisin $\mathbf{q}_j$ dans $\mathcal{Q}$. Le résultat de convergence de l'optimisation dépend évidemment de l'exactitude de cette approximation.

La difficulté du problème provient du fait que, peu importe l'approximation de la fonction de distance utilisée, la fonction d'erreur présentée à l'équation (2.6) n'est en général pas convexe [10]. Ce qui rend cette fonction non convexe est le fait que, pour chaque valeur de $\mathbf{R}$ et $\mathbf{t}$ données, la région de $\mathbf{\Phi}$ associée à un point $\mathbf{p}_i$, utilisée pour le calcul de la distance, change. Un algorithme d'optimisation basé sur une descente du gradient (ou l'équivalent), comme il sera présenté à la section 2.3.1, ne peut par conséquent pas garantir la convergence vers le minimum global. De plus, pour des nuages de points en trois dimensions, la transformation géométrique possède six degrés de liberté, soient trois rotations et trois translations. Il est impossible en pratique, à cause des limitations de calculs, d'explorer l'espace de recherche en entier. Des stratégies plus complexes et adaptées au problème, comme il sera présenté à la section 2.3.2, doivent donc être employées.

Par ailleurs, afin qu'une solution unique et non ambigüe à l'équation (2.5) existe, des hétérogénéités doivent être présentes dans l'environnement. Ce sont en effet ces dernières qui permettent de distinguer un bon calage d'un mauvais. Pour la suite, on suppose que l'effet de parallaxe est négligeable par rapport aux changements (hétérogénéités) dans l'environnement. On suppose également que l'environnement est statique, c'est-à-dire qu'il n'y a pas d'objets en déplacement. Même soumis à ces hypothèses de réduction, le problème s'applique à de

10. Voir `http://fr.wikipedia.org/wiki/Fonction_convexe`.

nombreuses situations en pratique. Lorsque ces hypothèses ne sont pas respectées, ce qui peut parfois être le cas, le problème devient plus difficile.

Sur une note additionnelle, le calage peut être réalisé par paires de nuages de points, tel que présenté ci-haut, ou encore simultanément sur un ensemble de nuages de points. Cette dernière approche permet d'optimiser globalement les transformations $\mathbf{R}$ et $\mathbf{t}$ entre chaque paire de nuage de points. Cela est particulièrement utile lorsque le robot retourne à un endroit déjà visité, puisque cette situation ajoute la contrainte que tous les nuages de points correspondants à cette région doivent s'aligner au même endroit. Les méthodes pour résoudre ce problème global ne seront toutefois pas présentées.

2.3.1 Algorithme *Iterative Closest Point*

Proposé par Besl et McKay [3] en 1992, l'algorithme *Iterative Closest Point* (ICP) est couramment utilisé en robotique mobile pour caler deux nuages de points. Celui-ci effectue une descente du gradient pour minimiser un cas particulier de l'équation (2.6). Il est donc simple et rapide, mais ne garantit pas de trouver la solution optimale ; il réalise un *calage local*. Pour cette raison, ICP est souvent utilisé dans une deuxième étape du calage, la première étant un *calage grossier* (ou *calage global*) permettant d'abord d'aligner approximativement les nuages de points. Idéalement, cet alignement initial se retrouve dans le bassin de convergence du minimum global de la fonction d'erreur (minimisée par ICP). ICP permet donc de raffiner l'alignement obtenu par le calage grossier, ce qui permet aussi de dire qu'il réalise le *calage fin*.

Tel que proposé originellement, l'approximation de la fonction de distance utilisée est la distance euclidienne (au carré) au point le plus proche (PPV de $\mathbf{p}_i$ dans $\mathcal{Q}$). Les points de $\mathcal{P}$ sont donc d'abord associés à leur plus proche voisin dans $\mathcal{Q}$, puis l'erreur est minimisée en fonction de ces associations pour évaluer $\mathbf{R}$ et $\mathbf{t}$. La transformation géométrique obtenue est ensuite utilisée pour aligner les nuages de points. Sachant que cette approximation de la fonction de distance est inexacte, ce processus est répété itérativement, jusqu'à ce qu'un critère d'arrêt soit atteint. Les critères d'arrêt possibles sont un nombre maximal d'itérations, un seuil maximal sur l'erreur et un seuil minimal sur la rotation et la translation. Plus formellement, l'algorithme est le suivant.

Cette version de ICP est connue sous le nom de *point-to-point* ICP. L'approximation de la fonction de distance d^2 par la distance euclidienne entre $\mathbf{p}_i$ et $\mathbf{q}_{\mathbf{p}_i}$ n'est valide que lorsqu'un point $\mathbf{p}_i$ est situé loin de la surface $\mathbf{\Phi}_\mathcal{Q}$. Pour les points situés proches, cette approximation est beaucoup moins précise. Une variante de l'algorithme, appelée *point-to-plane* ICP, se base plutôt sur la distance euclidienne (au carré) entre $\mathbf{p}_i$ et le plan tangent à la surface $\mathbf{\Phi}_\mathcal{Q}$ au point $\mathbf{q}_{\mathbf{p}_i}$ [11]. Cette variante est moins précise que la précédente pour les points situés loin de la surface, mais l'est davantage pour les points situés proche. Ses performances sont par

11. Le plan tangent réel étant inconnu (surface réelle inconnue), celui-ci doit être approximé.

Algorithm 1 Iterative Closest Point

while Aucun critère d'arrêt n'est atteint **do**

1. Associer les points de $\mathcal{P}$ à ceux de $\mathcal{Q}$ selon un critère du plus proche voisin.
2. Évaluer $\mathbf{R}$ et $\mathbf{t}$ minimisant la fonction d'erreur quadratique moyenne (convexe)

$$\varepsilon(\mathbf{R}, \mathbf{t}) = \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} || \left(\mathbf{R}\mathbf{p}_i + \mathbf{t}\right) - \mathbf{q}_{\mathbf{p}_i} ||^2 \tag{2.7}$$

   où $\mathbf{q}_{\mathbf{p}_i}$ est le point du nuage $\mathcal{Q}$ associé au point $\mathbf{p}_i$ du nuage $\mathcal{P}$.
3. Appliquer, selon l'équation (2.4), la transformation géométrique $\mathbf{R}$ et $\mathbf{t}$ aux points de $\mathcal{P}$.

end while

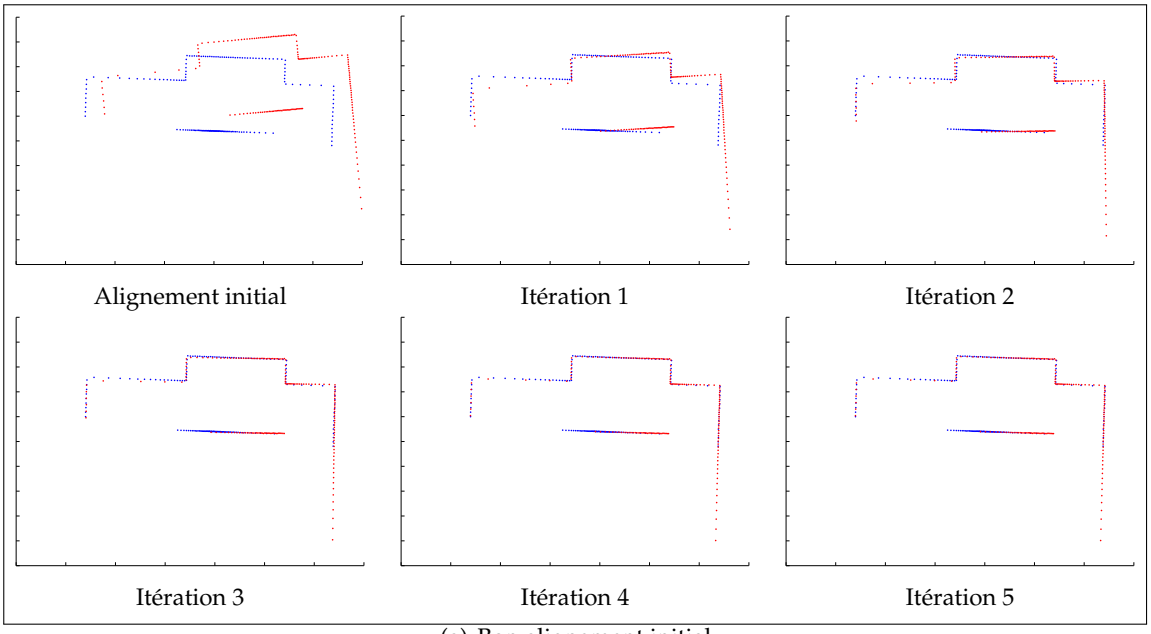
conséquent meilleures pour le calage fin. Une autre approximation est développée dans [25] et se révèle être une généralisation des deux précédentes. Cette dernière permet d'approximer plus précisément autant les points situés loin de la surface que ceux situés proche de celle-ci.

Plusieurs autres variantes de l'algorithme ont été proposées dans la littérature afin d'en améliorer les performances. Outre la mesure de distance utilisée, il est entre autres possible d'utiliser un autre critère que celui du plus proche voisin pour associer les points. L'article [32] fait une synthèse des variantes et compare leurs performances.

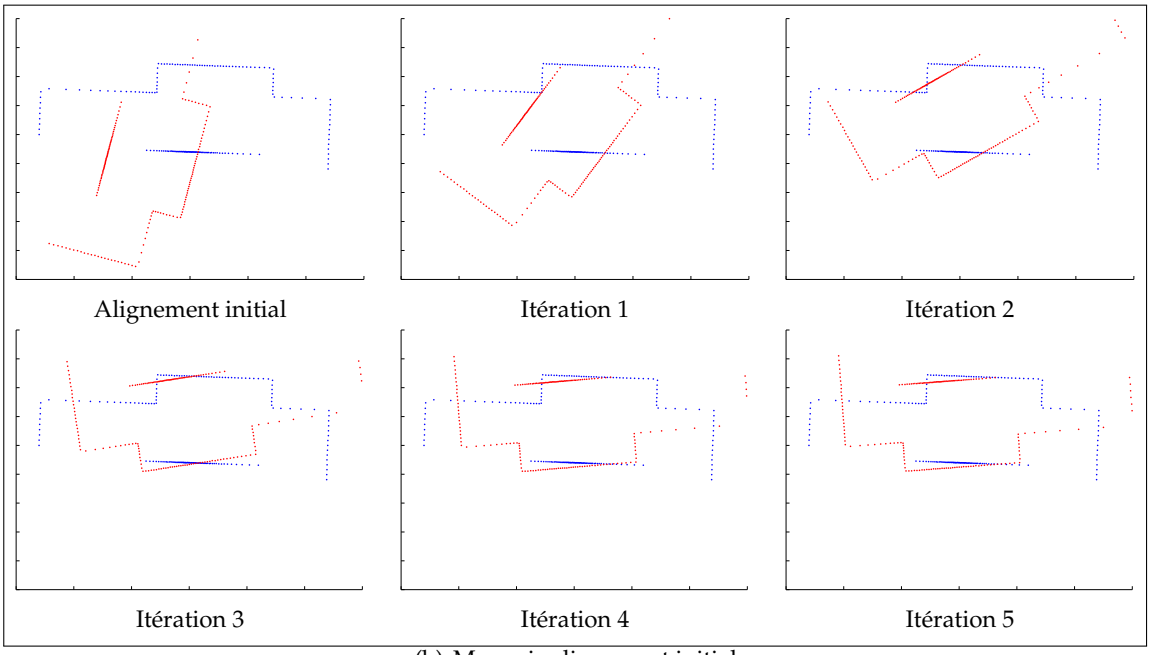
Tel qu'expliqué plus haut, peu importe la variante employée, ICP ne peut pas garantir de trouver la solution globale, car il converge toujours vers un minimum local. Pour mener à un bon calage, les nuages de points doivent être relativement bien alignés au départ. À titre d'exemple, la Figure 2.6 présente la progression de la convergence de ICP au fil des itérations pour deux alignements initiaux de différentes qualités. Dans le cas du bon alignement initial, l'algorithme converge très rapidement vers la bonne solution (après 3 itérations). Dans le cas du mauvais alignement initial, l'algorithme ne converge pas vers la vraie solution. L'optimisation est dans ce cas prise dans un minimum local. Cet exemple démontre l'importance du calage grossier.

2.3.2 Calage global

La raison principale pour laquelle le problème de calage est difficile est que les vraies associations entre les points de $\mathcal{P}$ et ceux de $\mathcal{Q}$ (ou la surface $\mathbf{\Phi}_\mathcal{Q}$) sont inconnues. En effet, le nombre total d'associations possibles croît exponentiellement en fonction de la taille des nuages de points. Une fois les points associés (comme à l'équation (2.7)), le problème devient convexe et facile à résoudre. Un algorithme comme ICP n'est pas certain de mener à la solution optimale, car ces associations sont approximées. Comme le résultat de la convergence dépend directement de ces associations, cet algorithme reste facilement coincé dans un minimum local et ce, surtout lorsque l'écart d'alignement entre les nuages de points est grand au départ (premières



(a) Bon alignement initial.



(b) Mauvais alignement initial.

FIGURE 2.6: Exemples de convergence de ICP au fil des itérations en fonction de la qualité de l'alignement initial.

associations erronées).

Par conséquent, une solution pour réaliser le calage global consiste à développer des méthodes plus précises pour établir des associations entre les nuages de points. Il s'agit du problème

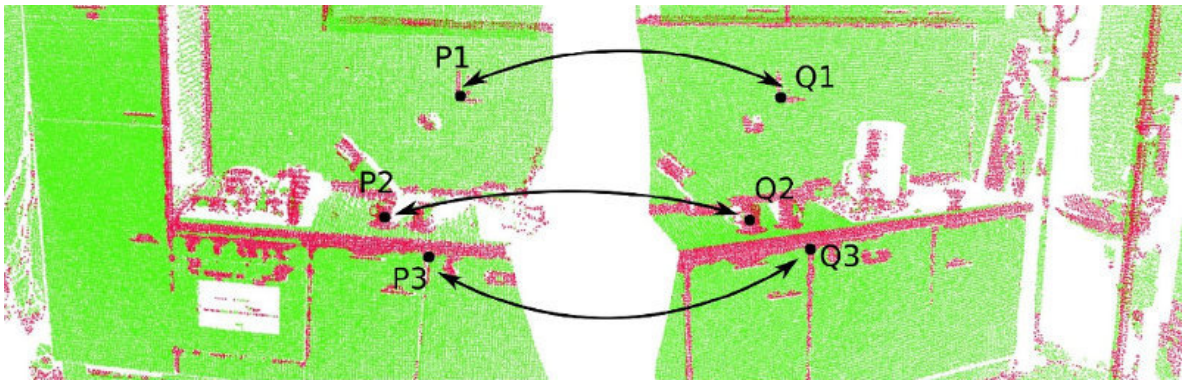


FIGURE 2.7: Exemples de correspondances valides établies entre deux nuages de points en comparant des descripteurs. Image tirée de [37].

d'*estimation de correspondances*. L'idée est de reconnaître des régions (objets) semblables entre les nuages de points, qui peuvent alors être associées les unes aux autres. Une façon d'y parvenir est de représenter des caractéristiques géométriques des points à l'aide de *descripteurs* afin de pouvoir les comparer. Ainsi, des points dont les descripteurs sont semblables peuvent être considérés comme potentiellement correspondants. La notion de descripteurs sera détaillée à la section 2.4. La Figure 2.7 montre des exemples de correspondances établies entre deux nuages de points en comparant des descripteurs.

Des stratégies différentes pour réaliser le calage global existent, mais ne seront pas abordées. Cette section présente deux méthodes, toutes deux basées sur l'algorithme *Random Sample Consensus* [14]. Cet algorithme, dont la compréhension est nécessaire pour la suite, est d'abord présenté. La première méthode de calage adapte *Random Sample Consensus* au problème de calage de nuages de points. La deuxième est une adaptation plus poussée et intègre l'utilisation des descripteurs.

Algorithme *Random Sample Consensus*

Random Sample Consensus (RANSAC) [14] est un algorithme stochastique permettant de caler un modèle mathématique $\mathcal{M}(\boldsymbol{\theta})$, où $\boldsymbol{\theta}$ est l'ensemble des paramètres du modèle, sur un ensemble de données $\mathcal{X}$. Contrairement aux méthodes analytiques, il permet d'estimer les paramètres $\boldsymbol{\theta}$ menant à la meilleure représentation des données possible même lorsque les données sont contaminées avec une grande quantité de bruit et de données aberrantes.

L'algorithme fonctionne par itérations, au cours desquelles des données sont pigées aléatoirement et utilisées pour déterminer des paramètres potentiels $\boldsymbol{\theta}_i$ du modèle. L'ensemble $\mathcal{C}_i$ des données cohérentes avec le modèle $\mathcal{M}(\boldsymbol{\theta}_i)$ est déterminé en fonction d'une erreur de tolérance ϵ_{tol} entre les données et les valeurs prédites par le modèle. Lorsque le nombre (ou la proportion) de données cohérentes est supérieur à un seuil c_{min}, la précision du modèle peut être améliorée en recalculant ses paramètres, cette fois en utilisant l'ensemble des données

cohérentes. Le processus est répété pour un nombre d'itérations N_{iter} donné et le modèle menant à l'erreur de représentation la plus faible est conservé. L'algorithme est présenté ici-bas.

Algorithm 2 RANSAC

for $i = 1 \rightarrow N_{iter}$ **do**
  $\mathcal{S} \leftarrow$ Sélectionner aléatoirement s_{min} données dans $\mathcal{X}$.
  $\boldsymbol{\theta}_i \leftarrow$ Déterminer les paramètres du modèle $\mathcal{M}(\boldsymbol{\theta}_i)$ en utilisant $\mathcal{S}$.
  $\mathcal{C}_i \leftarrow$ Déterminer les données cohérentes avec $\mathcal{M}(\boldsymbol{\theta}_i)$ en fonction de ϵ_{tol}.
  if $|\mathcal{C}_i| \geq c_{min}$ **then**
    $\boldsymbol{\theta}_i \leftarrow$ Réestimer les paramètres $\boldsymbol{\theta}_i$ en utilisant l'ensemble $\mathcal{C}_i$.
    $\varepsilon_i \leftarrow$ Évaluer la qualité du modèle par une fonction d'erreur.
  end if
end for
return Les paramètres $\boldsymbol{\theta}_i$ du modèle engendrant l'erreur ε_i la plus faible.

La nature aléatoire de RANSAC lui procure une très grande robustesse au bruit et aux données aberrantes. Un modèle valide (c.-à-d. représentant bien les données) est cependant obtenu avec une certaine probabilité. En effet, pour obtenir un tel modèle, toutes les données pigées aléatoirement au début d'une itération doivent être des données valides (non aberrantes). La probabilité de trouver un modèle valide au cours d'une itération est donc [12] :

$$p_{succès} = \left(\frac{|\mathcal{V}|}{|\mathcal{X}|}\right)^{s_{min}} \tag{2.8}$$

où $\mathcal{V}$ est l'ensemble des données valides parmi $\mathcal{X}$. Cette probabilité décroît lorsque le nombre de données aberrantes augmente. Il est par ailleurs habituellement préférable de piger le nombre minimum s_{min} de données requises pour déterminer les paramètres du modèle. Pour une ligne, par exemple, $s_{min} = 2$. Ce choix augmente la probabilité que toutes les données pigées soient valides. Pour illustrer l'influence de la présence de données aberrantes et l'impact du choix de la valeur de s_{min}, le Tableau 2.2 donne la valeur de $p_{succès}$ pour différentes combinaisons de valeurs. Enfin, le nombre espéré d'itérations à effectuer avant de trouver un modèle valide est :

$$E(N_{iter}) = \frac{1}{p_{succès}} = \left(\frac{|\mathcal{X}|}{|\mathcal{V}|}\right)^{s_{min}} \tag{2.9}$$

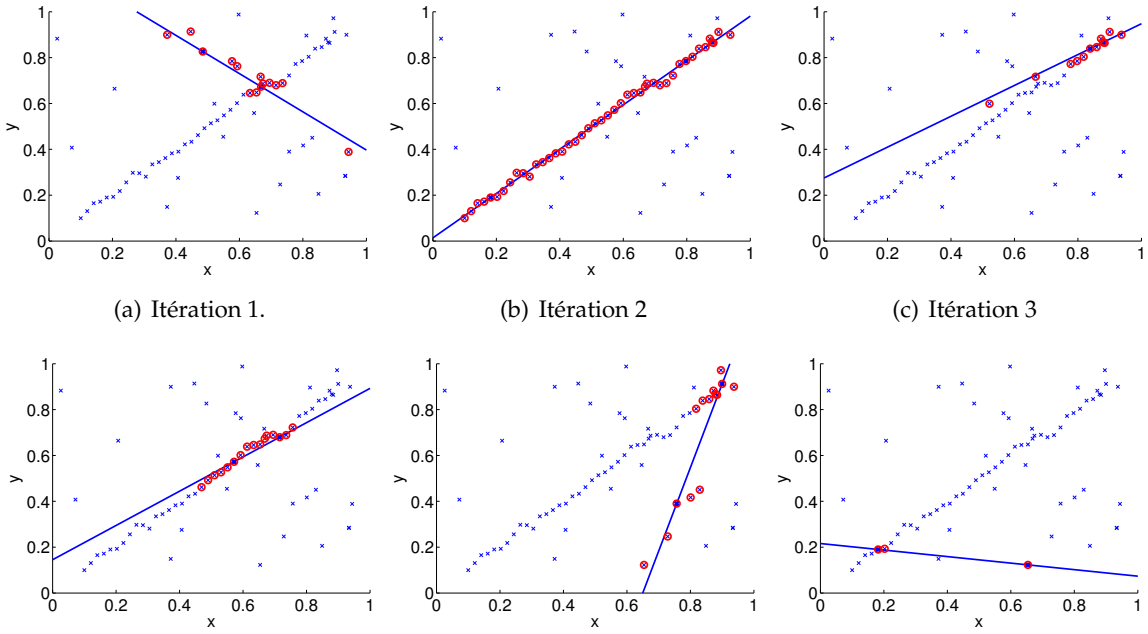
En présence de bruit dans les données, il est préférable d'obtenir plusieurs modèles valides et de conserver le meilleur. Il faut alors un plus grand nombre d'itérations.

La Figure 2.8 présente un exemple de résultat pour quelques itérations de RANSAC pour caler un modèle de droite sur des données en deux dimensions. Dans cet exemple, un bon modèle est obtenu à la 2^e itération.

12. Cette probabilité ne tient pas compte du fait que les données pigées doivent également être différentes et est par conséquent une approximation.

s_{min}	Proportion de données valides								
	0.9	**0.8**	**0.7**	**0.6**	**0.5**	**0.4**	**0.3**	**0.2**	**0.1**
2	0.81	0.64	0.49	0.36	0.25	0.16	0.09	0.04	0.01
3	0.73	0.51	0.34	0.22	0.13	0.064	0.027	0.008	0.001

TABLE 2.2: Probabilité de succès de RANSAC au cours d'une itération pour différentes proportions de données valides ($|\mathcal{V}|/|\mathcal{X}|$) et différentes valeurs de s_{min}.



(a) Itération 1. (b) Itération 2 (c) Itération 3

(d) Itération 4. (e) Itération 5 (f) Itération 6

FIGURE 2.8: Quelques itérations de l'algorithme RANSAC pour caler un modèle de droite sur un ensemble de données contaminées par du bruit et des données aberrantes en deux dimensions.

Calage avec *Random Sample Consensus*

Tel que présenté dans [15], RANSAC peut être adapté au problème de calage de nuages de points. On pose le nuage de points calé $\mathcal{P}_{calé} = \mathbf{R}\mathcal{P}_0 + \mathbf{t}$ (équation (2.4)) comme étant le modèle $\mathcal{M}(\boldsymbol{\theta})$, où $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{t}\}$. Les données à représenter sont celles du nuage de points $\mathcal{Q}$. L'algorithme 3 s'applique aux nuages de points en deux dimensions, mais est facilement extensible à trois dimensions. À chaque itération, une paire de points est sélectionnée aléatoirement dans $\mathcal{P}$ et dans $\mathcal{Q}$, avec comme contrainte que la distance entre les points doit être similaire dans les deux cas. L'hypothèse est faite que ces paires de points correspondent l'une à l'autre. Cette contrainte permet donc d'éliminer les cas où cette correspondance est impossible ou peu probable. Les paramètres $\mathbf{R}$ et $\mathbf{t}$ sont ensuite calculés en fonction de ces paires de points. Après un nombre d'itérations donné, le modèle menant au plus grand

nombre de points cohérents est réestimé en utilisant l'ensemble de ces points cohérents.

Algorithm 3 Calage de nuages de points 2D avec RANSAC

for $i = 1 \rightarrow N_{iter}$ **do**
- $\mathcal{S}_\mathcal{P} \leftarrow$ Sélectionner aléatoirement une paire de points $\{\mathbf{p}_1, \mathbf{p}_2\}$ dans $\mathcal{P}$ et calculer leur distance $d_\mathcal{P} = \|\mathbf{p}_1 - \mathbf{p}_2\|$.
- $\mathcal{S}_\mathcal{Q} \leftarrow$ Sélectionner aléatoirement une paire de points $\{\mathbf{q}_1, \mathbf{q}_2\}$ dans $\mathcal{Q}$ avec comme contrainte $d_\mathcal{Q} = \|\mathbf{q}_1 - \mathbf{q}_2\| \approx d_\mathcal{P}$, c'est-à-dire que $|d_\mathcal{P} - d_\mathcal{Q}| \leq d_{max}$ où d_{max} est un seuil défini par l'utilisateur. Si une telle paire de points n'existe pas, recommencer l'étape précédente.
- $\boldsymbol{\theta}_i \leftarrow$ Déterminer les paramètres $\mathbf{R}$ et $\mathbf{t}$ du modèle $\mathcal{M}(\boldsymbol{\theta}_i)$ en utilisant les paires de points sélectionnées $\mathcal{S}_\mathcal{P}$ et $\mathcal{S}_\mathcal{Q}$.
- $\mathcal{P}_{calé} \leftarrow$ Appliquer la transformation $\mathbf{R}$ et $\mathbf{t}$ au nuage de points $\mathcal{P}$.
- $\mathcal{C}_i \leftarrow$ Déterminer les données de $\mathcal{Q}$ cohérentes avec le modèle $\mathcal{M}(\boldsymbol{\theta}_i)$ en fonction de $\mathcal{P}_{calé}$, $\mathcal{Q}$ et d'une tolérance ϵ_{tol}.

end for

$\boldsymbol{\theta} \leftarrow$ Sélectionner l'hypothèse menant au nombre de données cohérentes $|\mathcal{C}_i|$ le plus grand et utiliser toutes les données cohérentes $\mathcal{C}_i$ pour réestimer $\mathbf{R}$ et $\mathbf{t}$ qui minimisent une fonction d'erreur.

return Les paramètres $\boldsymbol{\theta}$ du modèle.

Dans l'algorithme présenté, le nombre de points pigés à chaque itération est de deux dans chaque nuage de points, car c'est le minimum requis pour déterminer $\mathbf{R}$ et $\mathbf{t}$ en deux dimensions. Pour des nuages de points à trois dimensions, ce nombre serait de trois (non coplanaires). Il est pertinent de remarquer que, contrairement à l'algorithme RANSAC décrit plus haut, les paramètres du modèle ne sont pas réestimés chaque fois que le nombre de points cohérents dépasse un seuil, mais uniquement à la fin. Il existe en effet différentes variantes de RANSAC à cet égard.

Calage avec *Sample Consensus Initial Alignment*

L'algorithme *Sample Consensus Initial Alignment* (SAC-IA) [34] est un algorithme de calage lui aussi basé sur la philosophie de RANSAC. Il tire profit de l'utilisation des descripteurs pour établir aléatoirement, mais plus stratégiquement, des correspondances entre les nuages de points.

Une itération débute par la pige aléatoire de s_{min} points (trois en 3D [13]) dans le nuage de points $\mathcal{P}$, avec comme contrainte que leurs distances relatives (entre chaque paire de points) doit être supérieure à un seuil d_{min}. Ensuite, pour chacun de ces points $\mathbf{p}_i$, la liste des points du nuage $\mathcal{Q}$ dont les descripteurs sont les plus semblables à celui de $\mathbf{p}_i$ (k-PPV dans l'espace des caractéristiques) est construite. Noter que le concept de similarité entre les descripteurs

13. En trois dimensions, trois contraintes (non colinéaires) suffisent pour déterminer la rotation et la translation d'un objet. On pose donc $s_{min} = 3$.

est détaillé à la section suivante. Le point de $\mathcal{Q}$ correspondant au point $\mathbf{p}_i$ est sélectionné aléatoirement parmi les points de cette liste. Ceci permet de ne considérer que les correspondances probables et se révèle donc beaucoup plus performant que la méthode précédente. Les paramètres **R** et **t** sont ensuite calculés et évalués à l'aide d'une fonction d'erreur. Dans cet algorithme, il n'y a pas de décompte du nombre de points cohérents. La qualité du modèle est évaluée directement à chaque itération.

Algorithm 4 Sample Consensus Initial Alignment

for $i = 1 \rightarrow N_{iter}$ **do**
- $\mathcal{S}_\mathcal{P} \leftarrow$ Sélectionner aléatoirement s_{min} points dans $\mathcal{P}$ tels que leurs distances relatives (entre chaque paire) est supérieure à d_{min}.
- $\mathcal{S}_\mathcal{Q} \leftarrow$ Pour chaque point sélectionné $\mathbf{p}_i$, établir une liste des k points dans $\mathcal{Q}$ dont les descripteurs sont les plus semblables à celui de $\mathbf{p}_i$ et piger aléatoirement un point $\mathbf{q}_i$ dans cette liste.
- $\boldsymbol{\theta}_i \leftarrow$ Déterminer les paramètres **R** et **t** du modèle $\mathcal{M}(\boldsymbol{\theta}_i)$ en utilisant les points sélectionnés $\mathcal{S}_\mathcal{P}$ et $\mathcal{S}_\mathcal{Q}$.
- $\varepsilon_i \leftarrow$ Évaluer la qualité du modèle par une fonction d'erreur (équation (2.6)).

end for

return Les paramètres $\boldsymbol{\theta}_i$ du modèle engendrant l'erreur ε_i la plus faible.

Le lecteur pourra se questionner sur la raison pour laquelle la correspondance d'un point $\mathbf{p}_i$ n'est pas directement établie avec le point de $\mathcal{Q}$ dont le descripteur est le plus semblable, mais plutôt sélectionné aléatoirement dans la liste des k points les plus semblables. En fait, le point $\mathbf{q}_i$ réellement correspondant à un point $\mathbf{p}_i$ n'est pas nécessairement le plus semblable selon une comparaison des descripteurs. À cause, entre autres, du bruit de mesure, ce peut être par exemple le deuxième ou le troisième plus semblable. Cette stratégie permet d'explorer plus de possibilités, améliorant ainsi la robustesse de l'algorithme au bruit et à l'échantillonnage variable des surfaces selon la position du robot. Une valeur de k trop grande réduit cependant la probabilité de piger des correspondances valides et force par conséquent à augmenter le nombre d'itérations N_{iter}.

2.4 Descripteurs

Cette section détaille d'abord la notion de descripteurs, portant sur l'extraction et la représentation de caractéristiques dans les nuages de points dans le but de reconnaître certaines régions ou objets. La sous-section 2.4.2 explique comment les descripteurs peuvent être utilisés pour estimer des correspondances. La sous-section 2.4.3 présente ensuite quelques types de descripteurs publiés dans la littérature.

2.4.1 Définition d'un descripteur

Un descripteur est un ensemble de valeurs caractéristiques, attribué à un point $\mathbf{p}$, décrivant la géométrie locale autour de ce point. Ces valeurs sont calculées à partir des coordonnées (x, y, z) de $\mathbf{p}$ et de celles de ses voisins. Un descripteur peut donc être représenté sous forme de vecteur par

$$\mathbf{f} = \begin{bmatrix} f_1 & f_2 & \dots & f_D \end{bmatrix} \tag{2.10}$$

où D est la dimensionnalité du descripteur (nombre de dimensions). Les descripteurs peuvent être calculés systématiquement pour chaque point dans un nuage de points, ou pour un sous-ensemble de points préalablement sélectionnés. À cet effet, certaines méthodes de calcul de descripteurs incorporent une étape d'*extraction de points d'intérêt* pour lesquels sont ensuite calculées les valeurs de description.

Idéalement, un descripteur possède les caractéristiques suivantes : i) robuste ; ii) stable ; iii) unique et non ambigu ; iv) invariant aux rotations, aux translations et à la densité de points ; v) discriminant. La robustesse du descripteur signifie que ses valeurs ne changent pas ou très peu en présence de bruit de mesure. La stabilité s'applique à l'extraction des points d'intérêt uniquement, lorsque applicable, et signifie que la localisation des points sélectionnés ne varie pas ou très peu en présence de bruit. L'unicité et la non ambiguïté sont importantes afin que le calcul d'un descripteur, dans les mêmes conditions, mène toujours au même résultat. L'invariance aux rotations et aux translations implique que le descripteur d'un point est le même peu importe le point de vue. Il s'agit d'une caractéristique primordiale pour qu'une même région soit reconnue entre des nuages de points acquis à des endroits différents. Enfin, le pouvoir discriminant est ce qui rend possible la distinction des descripteurs de points situés sur des surfaces différentes. Par exemple, les valeurs calculées pour un point situé sur une surface plane ne devraient pas être les mêmes que celles obtenues pour un point situé sur un coin de mur. Cette caractéristique témoigne donc de la capacité à décrire avec précision la géométrie locale autour d'un point. D'autres caractéristiques intéressantes d'un descripteur sont sa rapidité de calcul et sa faible dimensionnalité.

La robustesse et le pouvoir discriminant d'un descripteur font habituellement l'objet d'un compromis. Un descripteur très discriminant est en général plus sensible au bruit, car il contient beaucoup d'informations précises, et inversement. Ce compromis amène à introduire deux catégories de descripteurs : les *signatures* et les *histogrammes*. Pour un point $\mathbf{p}$, les signatures encodent, par rapport à un repère local défini à $\mathbf{p}$, une ou plusieurs mesures géométriques calculées pour chacun des points $\mathbf{p}_k$ dans son voisinage, de même que la localisation de ces mesures. Les signatures sont donc potentiellement très discriminatives étant donnée la quantité d'informations précises qu'elles contiennent. Les histogrammes accumulent quant à eux ces mesures dans des classes, ce qui engendre une certaine perte d'information au profit d'une meilleure robustesse.

2.4.2 Estimation de correspondances à l'aide de descripteurs

Pour pouvoir effectuer des correspondances avec des descripteurs, il faut d'abord définir une mesure de distance entre les vecteurs $\mathbf{f}$. Le descripteur le plus semblable à un descripteur donné peut ensuite être obtenu par une recherche du plus proche voisin, dans l'espace des caractéristiques, selon cette métrique distance.

Différentes stratégies peuvent être adoptées pour établir des correspondances. Une approche simple basée sur le plus proche voisin consiste, pour le descripteur $\mathbf{f_p}$ d'un point $\mathbf{p} \in \mathcal{P}$, à trouver son plus proche voisin $\mathbf{f_q}$ parmi l'ensemble des descripteurs $\mathcal{F}_\mathcal{Q}$ du nuage de points $\mathcal{Q}$, puis à pairer les points $\mathbf{p}$ et $\mathbf{q}$ si la distance entre $\mathbf{f_p}$ et $\mathbf{f_q}$ est inférieure à un seuil (descripteurs suffisamment semblables), c'est-à-dire si $\|\mathbf{f_p} - \mathbf{f_q}\| < t_{corr}$. Comme il existe toujours un plus proche voisin (même pour des descripteurs d'une région observée dans $\mathcal{P}$ qui n'est pas présente dans $\mathcal{Q}$), le seuil permet d'éviter d'établir des correspondances peu probables lorsque les descripteurs sont trop différents. Une autre méthode similaire établit une correspondance lorsque le ratio entre la distance au premier et au second plus proche voisin est sous un seuil, c'est-à-dire si $\|\mathbf{f_p} - \mathbf{f_{q,1}}\| / \|\mathbf{f_p} - \mathbf{f_{q,2}}\| < t_{corr}$, où $\mathbf{f_{q,1}}$ et $\mathbf{f_{q,2}}$ sont le premier et le deuxième plus proches voisins respectivement.

Le nombre de correspondances établies entre deux ensembles de descripteurs dépend par conséquent fortement de la valeur de t_{corr}, qui agit comme un seuil de confiance sur les correspondances à établir. Plus la valeur de t_{corr} est petite, plus deux descripteurs doivent être similaires afin d'établir une correspondance. Le nombre de correspondances final est alors plus faible, mais la probabilité que chacune d'entre elles soit valide est plus élevée. À l'inverse, plus la valeur de t_{corr} est grande, plus l'estimation de correspondances est permissive sur les dissimilarités entre les descripteurs. Il y a dans ce cas un plus grand nombre de correspondances établies, mais la probabilité de chacune d'être valide est plus faible.

Pour quantifier les performances de l'estimation de correspondances, il existe, entre autres, deux métriques populaires : la *précision* et le *rappel*. La précision est définie comme étant la proportion de correspondances valides parmi toutes les correspondances établies, soit :

$$\text{précision} = \frac{\sharp \text{ corr. valides établies}}{\sharp \text{ corr. établies}} \tag{2.11}$$

Le rappel quant à lui est défini comme étant le nombre de correspondances valides trouvées divisé par le nombre total de correspondances valides possibles, c'est-à-dire :

$$\text{rappel} = \frac{\sharp \text{ corr. valides établies}}{\sharp \text{ corr. valides possibles}} \tag{2.12}$$

Idéalement, ces valeurs sont toutes deux très près de un, soit la valeur maximale. En pratique, elles font l'objet d'un compromis, contrôlé par la valeur du seuil de correspondance t_{corr}. En continuité avec ce qui a été mentionné ci-haut, le fait d'augmenter la valeur de t_{corr} mène en général à un rappel plus élevé, mais une précision plus faible.

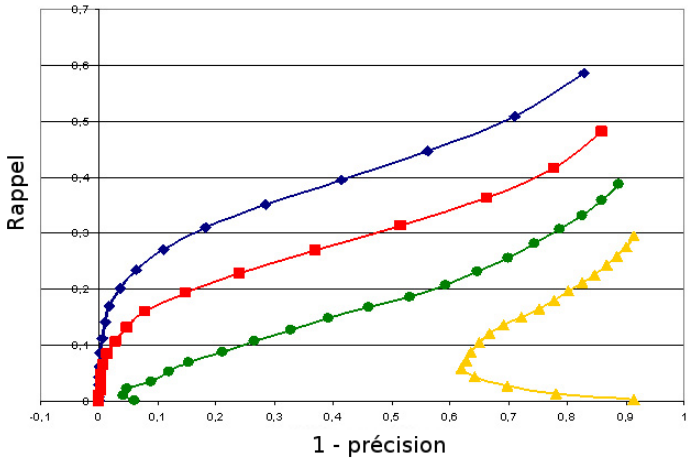


FIGURE 2.9: Exemples de courbes de précision et rappel. Image tirée de [46].

Une approche permettant de déterminer une valeur adéquate de t_{corr} est de tracer, pour un type de descripteur et un style d'environnement ou d'objet scanné donnés, une *courbe de précision et rappel*. Une telle courbe, traçant le rappel en fonction de la précision, est obtenue en estimant les correspondances entre deux scans pour différentes valeurs de t_{corr}. On peut ensuite utiliser cette courbe pour choisir une valeur de t_{corr} offrant un bon compromis entre la précision et le rappel. Le point d'opération optimal pour la valeur de t_{corr} peut par exemple être défini comme étant situé dans coude de la courbe. La Figure 2.9 présente des exemples de courbes ayant une forme typique, pour quatre types de descripteurs (non précisés) calculés sur des scans de statuettes. Ce graphique montre le rappel en fonction de $1 -$ précision, ce qui est commun. Le descripteur le plus performant dans ce cas est donc celui correspondant à la courbe bleue. Il faut préciser que tracer ces courbes implique que les vraies correspondances entre les scans soient connues, ou encore de disposer d'une méthode permettant de valider les correspondances. Un concept apparenté à la courbe de précision et rappel est la courbe *Receiver Operating Characteristic* [14] (ROC), qui peut elle aussi être utilisée avec les descripteurs.

Enfin, il vaut la peine de mentionner que Mikolajczyk et Schmid [24] font une bonne synthèse des méthodes d'évaluation et de comparaison de la performance des descripteurs pour l'estimation de correspondances.

2.4.3 Quelques types de descripteurs

Les sous-sections qui suivent présentent, par ordre chronologique de publication, quelques principaux types de descripteurs présentés dans la littérature. Uniquement ceux applicables aux nuages de points 3D acquis de scanners laser sont considérés. Pour le lecteur intéressé,

14. Voir `http://en.wikipedia.org/wiki/Receiver_operating_characteristic`.

Tangelder et Veltkamp [45] présentent une revue de littérature plus exhaustive qui couvre les principaux descripteurs 3D jusqu'en 2008.

Variation de la courbure

La variation de la courbure à un point $\mathbf{p}$ peut être estimée de façon similaire à la normale, par une décomposition en valeurs propres et vecteurs propres du voisinage (voir section 2.2.2). En posant les valeurs propres $\lambda_1 \leq \lambda_2 \leq \lambda_3$, la variation de la courbure peut être approximée par le ratio entre la plus petite valeur propre et la somme de toutes les valeurs propres :

$$M_{cc} = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.13}$$

Il existe bien entendu d'autres approches de calcul. Toutefois, cette caractéristique seule est sensible au bruit et peu performante.

Moment invariants

Les *moment invariants* [38] capturent la distribution spatiale des points autour d'un point central $p_0 = (x_0, y_0, z_0)$. Pour une fonction continue $\rho(x, y, z)$, les moments 3D d'ordre $p + q + r$ sont définis par

$$m_{pqr} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q z^r \rho(x, y, z) dx dy dz \tag{2.14}$$

Le moment central, centré à p_0 est donné par

$$\mu_{pqr} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_0)^p (y - y_0)^q (z - z_0)^r \rho(x, y, z) dx dy dz \tag{2.15}$$

Cela permet de définir, en épargnant les détails, des moments invariants à la rotation et à la translation. Par exemple, trois moments invariants du deuxième ordre sont :

$$J_1 = \mu_{200} + \mu_{020} + \mu_{002} \tag{2.16}$$

$$J_2 = \mu_{200}\mu_{020} + \mu_{200}\mu_{002} + \mu_{020}\mu_{002} - \mu_{110}^2 - \mu_{101}^2 - \mu_{011}^2 \tag{2.17}$$

$$J_3 = \mu_{200}\mu_{020}\mu_{002} + 2\mu_{110}\mu_{101}\mu_{011} - \mu_{200}\mu_{011}^2 - \mu_{020}\mu_{101}^2 - \mu_{002}\mu_{110}^2 \tag{2.18}$$

Spin images

Les *spin images* [19] capturent la distribution radiale des points dans le voisinage d'un point $\mathbf{p}$. La normale $\mathbf{n}$ du point $\mathbf{p}$ est utilisée pour définir partiellement un système de coordonnées cylindriques local. Pour chaque voisin $\mathbf{p}_k$ de $\mathbf{p}$, on peut alors définir deux coordonnées. La première, une coordonnée radiale α_k, est donnée par la distance perpendiculaire de $\mathbf{p}_k$ à $\mathbf{n}$. La deuxième, une coordonnée d'élévation β_k, est donnée par la distance, positive ou négative, entre $\mathbf{p}_k$ et le plan défini par le vecteur $\mathbf{n}$ et passant par $\mathbf{p}$. Les coordonnées (α, β) de tous les points voisins de $\mathbf{p}$ sont accumulées dans un histogramme 2D. Le résultat peut être interprété

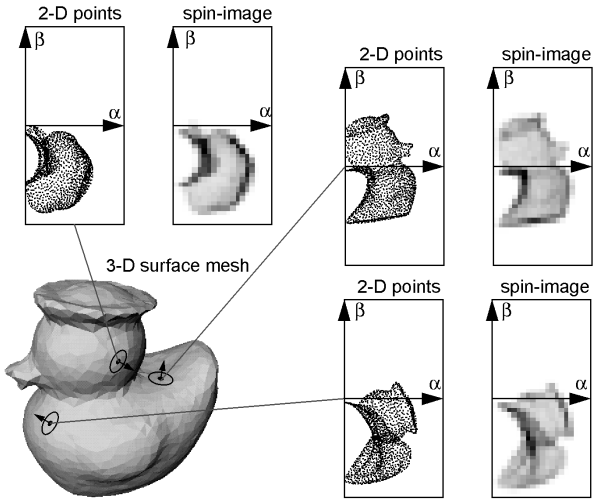


FIGURE 2.10: Exemples de descripteurs *spin images* calculés pour trois points sur un modèle de canard. Image tirée de [19].

comme une image, où l'intensité des pixels dépend du nombre de points accumulés dans la classe correspondante de l'histogramme. La Figure 2.10 montre les histogrammes obtenus, sous forme d'images, pour trois points d'un modèle de canard.

Point fingerprints

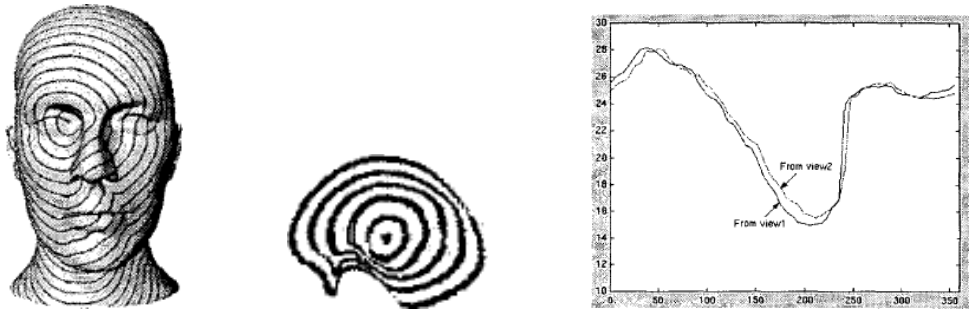
Les *point fingerprints* [44] sont des descripteurs encodant la variation de caractéristiques le long de cercles géodésiques [15] centrés sur un point $\mathbf{p}$. Ceux-ci sont calculés pour différentes valeurs de distance géodésique, puis projetés sur un plan tangent à la normale du point $\mathbf{p}$. Les contours obtenus par cette projection, similaires à ceux d'une emprunte digitale, sont utilisés pour encoder la variation des normales et de leur rayon par rapport au point central le long de chaque cercle géodésique. La Figure 2.11 illustre la méthode et présente un exemple de variation du rayon le long d'un cercle géodésique.

Shape context 3D

Les *shape contexts* [2], initialement proposés pour décrire des points sur des surfaces à deux dimensions, ont été étendus aux nuages de points 3D [16]. Ils représentent la distribution des points du voisinage d'un point $\mathbf{p}$ par la distribution des vecteurs reliant $\mathbf{p}$ à chacun de ses voisins $\mathbf{p}_k$. Un histogramme $\mathbf{h}$ des coordonnées relatives des voisins $\mathbf{p}_k$ par rapport à $\mathbf{p}$ est calculé selon :

$$\mathbf{h}(n) = \#\{(\mathbf{p}_k - \mathbf{p}) \in classe(n)\} \tag{2.19}$$

15. Un cercle géodésique est une ligne courbe fermée tout au long de laquelle la distance par rapport à son point central est constante suivant la surface.



(a) (b) (c)

FIGURE 2.11: Descripteur *point fingerprints*. a) Cercles géodésiques calculés autour d'un point sur un modèle de visage. b) Exemple de projection obtenue. c) Exemple de variation du rayon le long d'un cercle géodésique. L'axe des abscisses représente la position sur le contour (angle variant de 0 à 360 degrés). Images tirées de [44].

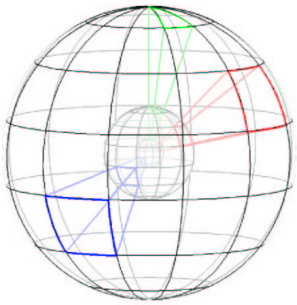


FIGURE 2.12: Illustration des classes de l'histogramme d'un *shape context 3D*, définies en coordonnées sphériques. Image tirée de [2].

où $n \in [1, N]$ est l'indice d'une classe de l'histogramme. Plus précisément, chacune des classes de l'histogramme correspond à une région de l'espace autour du point $\mathbf{p}$. Les voisins $\mathbf{p}_k$ situés dans une région donnée sont ajoutés à la classe correspondante. Les bornes des classes sont définies de sorte que, en coordonnées sphériques (θ, ϕ, r), leur répartition est uniforme en θ et en ϕ, et logarithmique en r, tel qu'illustré à la Figure 2.12. Ce choix rend le descripteur plus sensible aux points situés près de $\mathbf{p}$ qu'à ceux situés plus loin. Bien que le nombre de classes, N, soit paramétrable, celui-ci est habituellement élevé, ce que fait des *shape contexts 3D* des descripteurs à haute dimensionnalité.

L'invariance à la translation est intrinsèque à la méthode, puisque les données mesurées sont toujours relatives à $\mathbf{p}$. Pour obtenir l'invariance à la rotation, plutôt que d'utiliser les coordonnées globales des points, un repère local centré sur $\mathbf{p}$ est défini et utilisé pour construire l'histogramme. La définition de ce repère est toutefois en partie ambigüe. Sans apporter davantage de détails à ce problème, les *unique shape contexts* [47] y apportent une solution en définissant un repère unique et non ambigu.

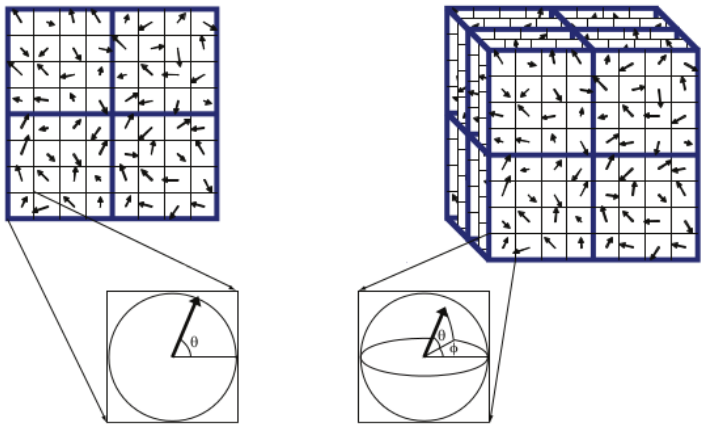


FIGURE 2.13: Illustration du descripteur SIFT (à gauche) et du SIFT 3D (à droite). Images tirées de [40].

SIFT 3D

Le descripteur *Scale-Invariant Feature Transform* (SIFT) [23], très utilisé en vision numérique, a été adapté aux données en trois dimensions [40]. Le descripteur résultant, le SIFT 3D, requiert toutefois une valeur supplémentaire aux coordonnées (x, y, z) associée à chaque point. Cette valeur, w, peut être par exemple la valeur d'intensité de retour du laser.

Brièvement, le calcul des SIFT s'effectue d'abord par une extraction de points d'intérêt. Pour une image, les points d'intérêt recherchés sont ceux où la valeur des pixels est un extremum stable. En 3D, les extremums sont déterminés en fonction de w. Les gradients de w sont ensuite calculés autour de chaque point d'intérêt, puis représentés sous forme d'histogramme. La Figure 2.13 illustre les gradients calculés et la structure du descripteur obtenu.

Point Feature Histograms

Plus récents, les *Point Feature Histograms* (PFH) [33] sont des descripteurs encodant beaucoup d'informations sur la géométrie locale autour d'un point $\mathbf{p}$. Ils sont à la fois très discriminants, robustes au bruit et invariants aux rotations, aux translations et à la densité des points. Les PFH encodent les variations angulaires des normales entre les points du voisinage de $\mathbf{p}$ et les accumulent dans un histogramme. Le calcul s'effectue de la manière suivante :

1. Sélectionner le voisinage de $\mathbf{p}$, délimité par une sphère de rayon r ;
2. Pour toute paire de points $\{\mathbf{p}_i, \mathbf{p}_j\}$ $(i \neq j)$ et leurs normales ($\mathbf{n}_i$ et $\mathbf{n}_j$) dans le voisinage de $\mathbf{p}$ (incluant le point $\mathbf{p}$ lui-même) :
   a) Choisir $\mathbf{p}_i$ tel que $\angle(\mathbf{n}_i, \mathbf{s}) \leq \angle(\mathbf{n}_j, \mathbf{s})$, où $\mathbf{s}$ est la droite reliant $\mathbf{p}_i$ et $\mathbf{p}_j$;
   b) Définir un repère de Darboux [16] $\mathbf{uvw}$ au point $\mathbf{p}_i$;

16. Repère mobile construit en un point donné d'une surface 3D et utilisé pour en décrire la courbure. Le triplet

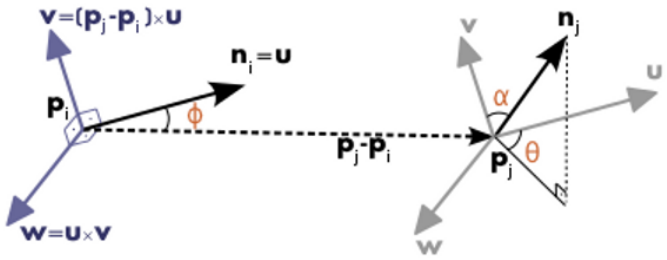


FIGURE 2.14: Illustration des variations angulaires calculées entre les normales de deux points lors du calcul d'un PFH. Image tirée de [37].

c) Calculer les variations angulaires, telles qu'illustrée à la Figure 2.14 entre $\mathbf{n}_i$ et $\mathbf{n}_j$:

$$\begin{aligned} \alpha &= \mathbf{v} \cdot \mathbf{n}_j \\ \phi &= \left(\mathbf{u} \cdot \left(\mathbf{p}_j - \mathbf{p}_i\right)\right) / \|\mathbf{p}_j - \mathbf{p}_i\| \\ \theta &= \arctan\left(\mathbf{w} \cdot \mathbf{n}_j, \mathbf{u} \cdot \mathbf{n}_j\right) \end{aligned} \tag{2.20}$$

ainsi que la distance entre les points $\mathbf{p}_i$ et $\mathbf{p}_j$:

$$d = \|\mathbf{p}_j - \mathbf{p}_i\| \tag{2.21}$$

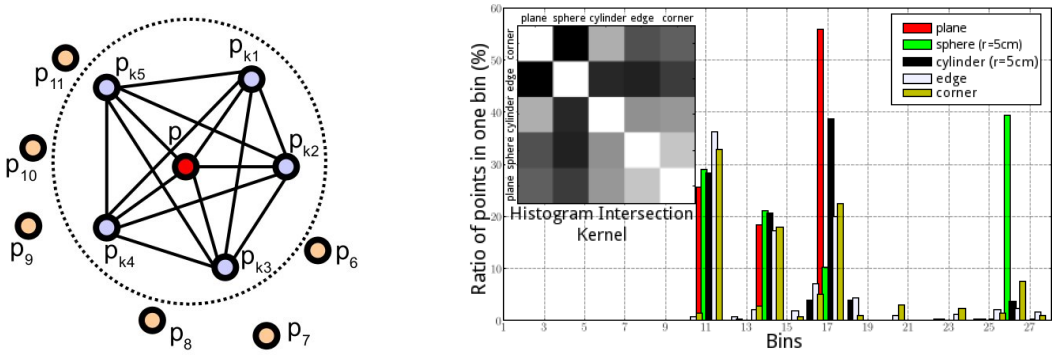
3. Construire un histogramme à N classes combinant tous les calculs pour le voisinage de $\mathbf{p}$, constitué successivement des classes correspondant à chacune des variables α, ϕ, θ et d.

La Figure 2.15(a) illustre la région d'influence (voisinage) pour le calcul du PFH d'un point $\mathbf{p}$. Le cercle pointillé représente la sphère délimitant le voisinage de $\mathbf{p}$. Les connexions entre les points marquent les paires de points contribuant au calcul du descripteur. La Figure 2.15(b) présente des exemples d'histogrammes obtenus pour différentes surfaces géométriques. La matrice « Histogram Intersection Kernel » affichée dans le coin supérieur gauche de l'image illustre une mesure de distance calculée entre les PFH des différentes surfaces. Plus une case est foncée, plus la distance est grande.

Fast Point Feature Histograms

Les *Fast Point Feature Histograms* (FPFH) [34] sont une amélioration des PFH. Ces derniers sont simplifiés et optimisés pour réduire significativement leur temps de calcul tout en préservant une grande partie de leur pouvoir descriptif. À titre informatif, la complexité algorithmique passe de $O\left(n \cdot k^2\right)$ pour les PFH à $O\left(n \cdot k\right)$ pour les FPFH, où n est le nombre de points dans le nuage et k le nombre de voisins de chacun des points (variable en fonction du rayon r délimitant le voisinage). Cette amélioration facilite leur utilisation en temps réel.

($\mathbf{u} = \mathbf{n}_i$, $\mathbf{v} = \left(\mathbf{p}_j - \mathbf{p}_i\right) \times \mathbf{u}$, $\mathbf{w} = \mathbf{u} \times \mathbf{v}$) forme une base orthonormale dont l'origine est le point choisi. Voir `http://en.wikipedia.org/wiki/Darboux_frame` pour plus de détails.



(a) Région d'influence.

(b) PFH de différentes surfaces géométriques.

FIGURE 2.15: Illustration de la région d'influence pour le calcul d'un PFH et exemples d'histogrammes obtenus pour différentes surfaces géométriques. Images tirées de [33].

La différence avec les PFH réside principalement dans le choix des paires de points utilisées pour le calcul des descripteurs. Plutôt que de considérer toutes les paires de points possibles dans le voisinage de $\mathbf{p}$, uniquement les paires $\{\mathbf{p}, \mathbf{p}_k\}$, où $\mathbf{p}_k$ est un voisin de $\mathbf{p}$, sont utilisées. Le FPFH final de $\mathbf{p}$ dépend toutefois du résultat obtenu, en suivant ce choix de paires de points, pour chacun de ses voisins. Cette simplification approxime le calcul du PFH utilisant toutes les paires de points. La seconde différence importante avec les PFH est le retrait de la distance entre les paires de points, d, du descripteur. Le calcul des FPFH s'effectue sur l'ensemble d'un nuage de points en deux phases comme suit :

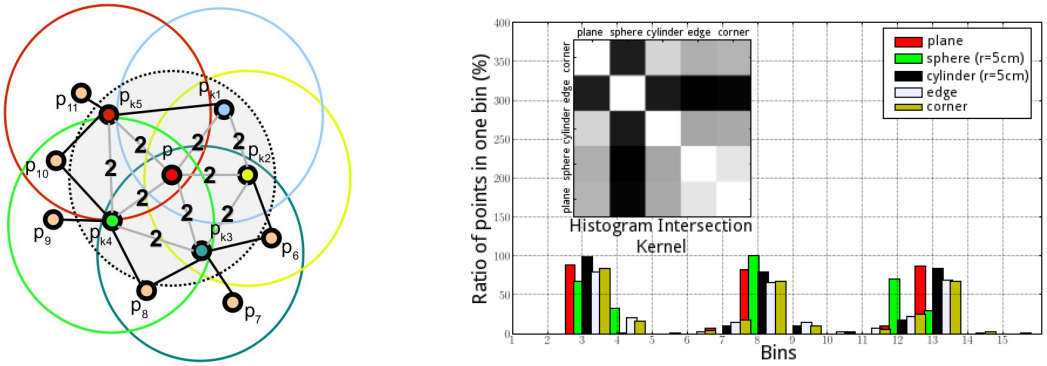
1. Pour tout point $\mathbf{p}$ et ses voisins $\mathbf{p}_k$, calculer, pour chaque paire de points $\{\mathbf{p}, \mathbf{p}_k\}$, les variations angulaires des normales selon l'équation (2.20). L'histogramme de $\mathbf{p}$ obtenu est un résultat intermédiaire appelé un *Simplified Point Feature Histogram* (SPFH).
2. Pour tout point $\mathbf{p}$, calculer son histogramme final en additionnant à son SPFH celui de chacun de ses voisins $\mathbf{p}_k$. Les SPFH des voisins sont pondérés en fonction de leur distance par rapport à $\mathbf{p}$:

$$FPFH(\mathbf{p}) = SPFH(\mathbf{p}) + \frac{1}{K} \sum_{k=1}^{K} w_k \cdot SPFH(\mathbf{p}_k) \tag{2.22}$$

où K est le nombre de voisins de $\mathbf{p}$ et $w_k = \dfrac{1}{\|\mathbf{p} - \mathbf{p}_k\|}$.

La Figure 2.16(a) illustre la région d'influence pour le calcul du FPFH d'un point $\mathbf{p}$. Le cercle grisé représente la sphère délimitant le voisinage de $\mathbf{p}$. Les autres cercles de couleur représentent celles des voisins $\mathbf{p}_k$. Contrairement aux PFH, les connexions ne relient pas toutes les paires de points possibles dans le voisinage de $\mathbf{p}$, illustrant le fait que toutes les paires ne contribuent pas au calcul. Les connexions marquées d'un 2 sont quant à elles considérées deux fois, soient une fois pour le SPFH de $\mathbf{p}$ et une fois pour le SPFH du voisin $\mathbf{p}_k$ correspondant.

La Figure 2.16(b) présente les histogrammes obtenus pour les mêmes surfaces géométriques qu'à la Figure 2.15(b). Certaines cases de la matrice « Histogram Intersection Kernel » sont cette fois plus pâles, illustrant une plus petite différence entre les descripteurs. Cela démontre une certaine diminution du pouvoir discriminant des FPFH par rapport aux PFH, bien qu'en pratique ils restent très performants.



(a) Région d'influence.

(b) FPFH de différentes surfaces géométriques.

FIGURE 2.16: Illustration de la région d'influence pour le calcul d'un FPFH et exemples d'histogrammes obtenus pour différentes surfaces géométriques autour d'un point **p**. Images tirées de [34].

Signature of Histograms of Orientations

Les *Signature of Histograms of Orientations* (SHOT) [46] sont un compromis entre les descripteurs de type signature et ceux de type histogramme. Les auteurs tentent de tirer les avantages de chacun de ces types tout en éliminant les inconvénients.

Pour le calcul du SHOT d'un point **p**, la sphère délimitant son voisinage, aussi appelée *région de support*, est divisée selon une grille sphérique isotropique, illustrée à la Figure 2.17. Pour chaque volume V_i de la grille, les voisins $\mathbf{p}_k$ appartenant à ce volume sont comptabilisés dans les classes d'un histogramme $\mathbf{h}_i$ selon la valeur retournée par la fonction $f(\theta_k) = \cos\theta_k$, où θ_k est l'angle formé par la normale $\mathbf{n}_k$ de $\mathbf{p}_k$ et la normale $\mathbf{n}$ de $\mathbf{p}$. Cette fonction est rapide à calculer car $\cos\theta_k = \mathbf{n}_k \cdot \mathbf{n}$. Un histogramme local est ainsi construit, pour chaque volume de la grille, et localisé par rapport à **p**. C'est cette localisation qui fait la similitude des SHOT avec les signatures. Le descripteur final du point **p** est l'ensemble des histogrammes locaux.

Pour localiser les histogrammes $\mathbf{h}_i$ par rapport à **p**, un repère de référence local est défini au point **p**. La définition précise de ce repère est cruciale pour obtenir l'invariance aux translations et aux rotations, de même qu'une bonne robustesse au bruit. Les auteurs des SHOT mettent une emphase particulière sur la définition d'un repère invariant, unique et non ambigu.

Par ailleurs, pour augmenter la robustesse au bruit, la comptabilisation des points dans les

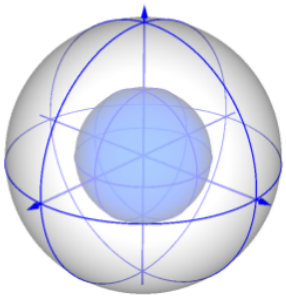


FIGURE 2.17: Illustration de la grille sphérique isotropique utilisée pour diviser la région de support pour le calcul des SHOT. Image tirée de [46].

histogrammes est pondérée selon la distance de chaque point par rapport au centre de son volume, de sorte que les points situés près des frontières aient moins de poids. Cela permet de réduire l'effet d'un point qui se retrouverait dans un autre volume de la grille à cause du bruit. Le descripteur final est également normalisé afin de le rendre invariant à la densité de points.

Chapitre 3

Notions d'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui concerne l'utilisation des mathématiques pour inférer, à partir d'observations, un modèle de traitement applicable à un certain type de données. Cela permet, entre autres, d'identifier des tendances dans les données ou de faire des prédictions par rapport à de nouvelles données. L'apprentissage automatique est particulièrement utile lorsque le modèle de traitement est a priori inconnu ou qu'il change dans le temps. Il existe plusieurs types d'apprentissage automatique. Ce chapitre se concentre spécifiquement sur l'apprentissage supervisé, car celui-ci sera employé dans ces travaux.

3.1 Apprentissage supervisé

L'apprentissage supervisé s'applique au cas particulier où les observations se présentent sous forme de paires $(\mathbf{x}, y)$, où $\mathbf{x}$ est un vecteur de valeurs d'entrée, appelées *caractéristiques* de l'observation, et y est une valeur scalaire de sortie qui y est associée :

$$(\mathbf{x}, y) = \left(\begin{bmatrix} x_1 & x_2 & ... \end{bmatrix}, y \right) \tag{3.1}$$

Soit un ensemble d'observations $\mathcal{X} = \{\mathbf{x}_t, y_t\}_{t=1}^{N}$, pour lesquelles la sortie y_t associée à chaque entrée $\mathbf{x}_t$ est connue. Cet ensemble est appelé l'ensemble d'*exemples*. L'apprentissage consiste à trouver, à partir de ces exemples, un modèle décrivant les associations entre les entrées $\mathbf{x}$ et les sorties y, de sorte qu'il soit ensuite possible de prédire les sorties y pour de nouvelles valeurs d'entrées $\mathbf{x}$ jamais observées au départ. Le modèle recherché peut être décrit par $y = h(\mathbf{x}|\boldsymbol{\theta})$, où h est le *prédicteur* et $\boldsymbol{\theta}$ est un vecteur représentant l'ensemble de ses paramètres. Le fonctionnement de l'apprentissage supervisé est illustré à la Figure 3.1. L'entraînement du prédicteur consiste à trouver les paramètres $\boldsymbol{\theta}$ qui optimisent un certain critère de performance (fonction d'erreur) afin de minimiser les erreurs de prédictions. On cherche par conséquent les

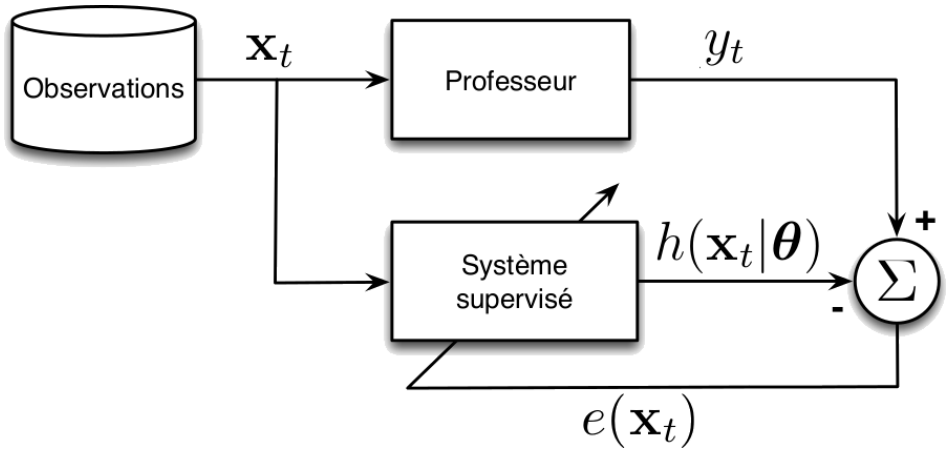


FIGURE 3.1: Illustration du fonctionnement de l'apprentissage supervisé. Image tirée de [17].

paramètres optimaux $\boldsymbol{\theta}^*$ du modèle qui minimisent une fonction d'erreur $\varepsilon(\boldsymbol{\theta}|\mathcal{X})$, c'est-à-dire :

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \varepsilon(\boldsymbol{\theta}|\mathcal{X}) \tag{3.2}$$

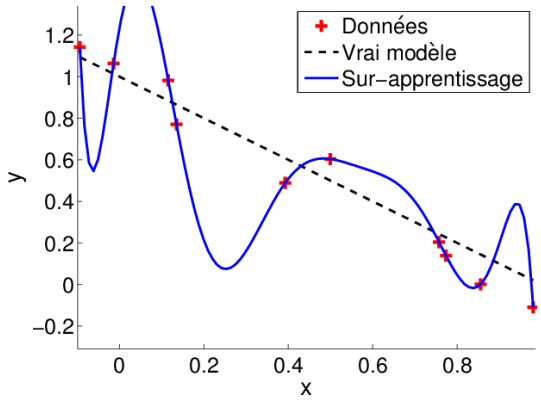
L'entraînement peut être défini de manière générale par la minimisation de l'erreur empirique

$$\varepsilon(\boldsymbol{\theta}|\mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N} \mathcal{L}\left(y_t, h(\mathbf{x}_t|\boldsymbol{\theta})\right) \tag{3.3}$$

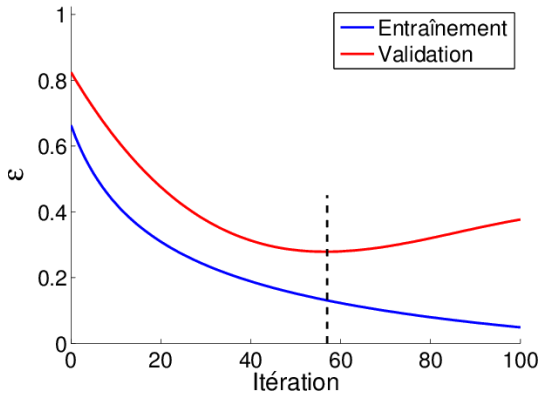
où $\mathcal{L}$ est une fonction de perte retournant une pénalité lorsque la prédiction du modèle n'est pas égale à la valeur réelle. Cette fonction de perte peut être par exemple :

$$\mathcal{L}(y, h(\mathbf{x}|\boldsymbol{\theta})) = \left(y_t - h(\mathbf{x}_t|\boldsymbol{\theta})\right)^2 \tag{3.4}$$

L'objectif d'un algorithme d'apprentissage est de produire un modèle minimisant le risque d'erreurs de prédictions sur de nouvelles données. Or, pour plusieurs algorithmes, la minimisation de l'équation (3.3) peut entraîner un sur-apprentissage : une situation où l'algorithme a « appris par coeur » les exemples d'entraînement. Le modèle optimisé décrit alors très bien les données vues en exemples, mais est plus complexe que le vrai modèle (inconnu) généralisant les données au-delà des exemples observés. Le prédicteur ainsi obtenu fait généralement peu d'erreurs sur les données utilisées pour l'entraînement, mais est peu performant sur de nouvelles données. Un exemple de sur-apprentissage est présenté à la Figure 3.2(a). Dans cet exemple, le vrai modèle ayant servi à générer les données est une droite, alors que le modèle optimisé est une courbe beaucoup plus complexe. Pour éviter le sur-apprentissage, certains algorithmes d'apprentissage incorporent un mécanisme de *régularisation* permettant de pénaliser les modèles complexes afin de favoriser l'apprentissage de modèles simples. Il est aussi possible d'utiliser un ensemble de données de *validation*, contenant des exemples différents de ceux utilisés pour l'apprentissage, afin de déterminer quand doit cesser l'optimisation.



(a) Exemple de prédicteur obtenu avec sur-apprentissage comparé au vrai modèle décrivant les données.



(b) Erreur en entraînement (bleu) et en validation (rouge). Le sur-apprentissage se traduit par une augmentation de l'erreur en validation. La limite en noir pointillé est le point optimal d'arrêt de l'optimisation.

FIGURE 3.2: Exemple de prédicteur obtenu en présence d'un sur-apprentissage et illustration de l'utilisation d'un ensemble de validation pour le détecter.

En effet, un début de sur-apprentissage se traduit par une augmentation du taux d'erreurs commises sur l'ensemble de validation. Ce phénomène est illustré à la Figure 3.2(b).

Pour quantifier le taux d'erreurs de prédictions, il faut donc distinguer le *risque empirique*, calculé à partir des exemples d'entraînement, du *risque réel* (pour de nouvelles données), qui est inconnu. Comme le démontre le sur-apprentissage, le risque empirique est biaisé et ne peut être utilisé pour évaluer les performances en généralisation du prédicteur. Typiquement, le risque réel est estimé en divisant aléatoirement l'ensemble des exemples $\mathcal{X}$ en un ensemble d'entraînement et un ensemble de test. L'ensemble d'entraînement est utilisé pour optimiser le modèle, alors que l'ensemble de test sert à quantifier les performances une fois l'entraînement terminé. Ainsi, aucun des exemples de test n'est utilisé pendant l'entraînement, ce qui permet d'effectuer une évaluation non biaisée du prédicteur. Pour certains algorithmes d'apprentissage, l'ensemble d'entraînement peut à son tour être divisé en sous-ensembles. Cela est utile, entre autres, lorsque des valeurs d'*hyperparamètres*, influençant le comportement de l'algorithme, doivent être déterminées. Ces sous-ensembles permettent de faire des essais d'entraînement à plus petite échelle, pour certaines valeurs d'hyperparamètres, et de quantifier les performances obtenues de manière non biaisée sans utiliser l'ensemble de test final.

Un cas particulier de l'apprentissage supervisé se présente lorsque la sortie y est une valeur discrète. Il s'agit alors du problème de *classification*, où l'objectif est de distinguer des données appartenant à différences classes (catégories) en fonction de leurs attributs. La suite de ce chapitre s'adresse uniquement au problème de classification de données.

3.1.1 Classification

Dans le problème de classification, la valeur de y associée à un exemple $\mathbf{x}$ est appelée son *étiquette* et représente la classe (catégorie) à laquelle appartient cette donnée. Le prédicteur h recherché, appelé dans ce cas-ci un *classificateur*, permet de faire des prédictions sur l'appartenance d'une donnée à l'une ou l'autre des classes.

Les algorithmes d'apprentissage pour la classification peuvent être séparés en deux catégories [29]. Les premiers optimisent des modèles dits *génératifs*, c'est-à-dire qu'ils estiment les densités de probabilités $p(\mathbf{x}|C_i)$ des données pour chacune des classes. L'hypothèse est faite que les données sont indépendantes et identiquement distribuées (iid) et qu'elles suivent une distribution, *a priori* inconnue. La classification est basée sur la probabilité estimée qu'une donnée $\mathbf{x}$ appartienne à chacune des classes :

$$h_i(\mathbf{x}) = \hat{P}(C_i|\mathbf{x}) = \hat{p}(\mathbf{x}|C_i)\hat{P}(C_i) \tag{3.5}$$

La classe prédite d'une donnée est donc celle dont la probabilité est la plus grande. Le modèle de densité de probabilité, $p(\mathbf{x}|C_i)$, peut être fixé à l'avance. Par exemple, l'hypothèse peut être faite que les données suivent une loi normale : $p(x|C_i, \boldsymbol{\theta}) = \mathcal{N}(\mu, \sigma^2)$, avec $\boldsymbol{\theta} = \{\mu, \sigma\}$. On parle alors d'une méthode paramétrique, dans laquelle les paramètres $\boldsymbol{\theta}$ doivent être optimisés. À l'inverse, lorsque aucune hypothèse n'est posée *a priori* sur la distribution des données, il s'agit d'une méthode non paramétrique.

La deuxième catégorie d'algorithmes d'apprentissage pour la classification comprend les méthodes qui résolvent uniquement le problème de discrimination, sans passer par l'estimation des densités de probabilité. Les modèles obtenus sont dits *discriminatifs*. Le prédicteur est alors une fonction discriminante $h_i(\mathbf{x}|\boldsymbol{\theta}_i)$. Une fonction discriminante permet de distinguer les données entre deux classes. Par conséquent, pour un problème à deux classes, un seul prédicteur suffit. Pour un problème à K classes, des solutions possibles sont, par exemple, d'entraîner un prédicteur h_i par classe ($i = 1, ..., K$) permettant de discerner les données de la classe C_i de toutes les autres, ou encore d'entraîner un prédicteur pour chaque paire de classes. La solution proposée au problème abordé plus loin dans ce mémoire fait appel à la classification à deux classes. Par conséquent, l'hypothèse d'un problème à deux classes est faite pour la suite de ce chapitre.

3.2 Quelques méthodes d'apprentissage

Cette section présente brièvement quelques algorithmes d'apprentissage supervisé qui peuvent être utilisés pour entraîner des classificateurs de type discriminatif.

3.2.1 Discriminant linéaire

Un discriminant linéaire est une équation décrivant un hyperplan séparateur, dans l'espace des caractéristiques, entre les données de deux classes $y \in \{C_1, C_2\}$. Pour des données à D dimensions (D caractéristiques), l'équation du prédicteur est donnée par

$$h(\mathbf{x}|\mathbf{w}, w_0) = \sum_{j=1}^{D} w_j x_j + w_0 = \mathbf{w}^T \mathbf{x} + w_0 \tag{3.6}$$

où le vecteur de coefficients $\mathbf{w}$ donne l'orientation de l'hyperplan et w_0 fixe sa position dans l'espace. La valeur retournée par ce classificateur est une valeur réelle. La classe prédite d'une donnée $\mathbf{x}$ est C_1 si $h(\mathbf{x}) \geq 0$ et C_2 si $h(\mathbf{x}) < 0$. L'hyperplan séparateur trace donc une *frontière de décision* (dans l'espace des caractéristiques) entre les deux classes.

Différentes méthodes permettent de calculer le modèle optimal d'un discriminant linéaire. L'algorithme du perceptron [1] minimise une fonction d'erreur équivalente à utiliser la fonction de perte

$$\mathcal{L}(y, h(\mathbf{x}|\mathbf{w}, w_0)) = \begin{cases} 1 & \text{si } y \neq h(\mathbf{x}|\mathbf{w}, w_0) \\ 0 & \text{si } y = h(\mathbf{x}|\mathbf{w}, w_0) \end{cases} \tag{3.7}$$

Cela revient à compter le nombre de données mal classées par $h(\mathbf{x}|\mathbf{w}, w_0)$. Comme l'erreur ne dépend pas de la distance entre les données et l'hyperplan, la convergence est atteinte pour une configuration quelconque du discriminant qui sépare les données. Par exemple, les deux discriminants linéaires obtenus aux Figures 3.3(a) et 3.3(b) sont équivalents. Cependant, pour des données non linéairement séparables, l'algorithme ne converge pas.

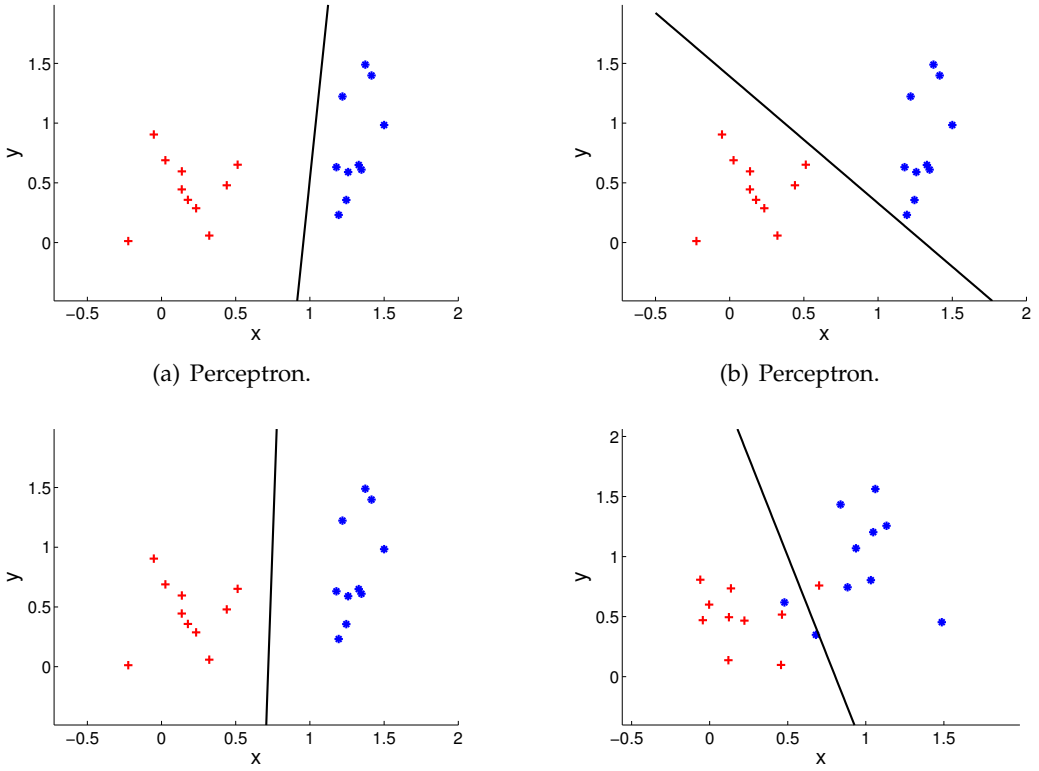
Une régression par la méthode des moindres carrés peut également être employée pour optimiser le discriminant. La fonction de perte utilisée dans ce cas est

$$\mathcal{L}(y, h(\mathbf{x}|\mathbf{w}, w_0)) = (y - h(\mathbf{x}|\mathbf{w}, w_0))^2 \tag{3.8}$$

L'erreur tient alors compte de la distance entre les données et l'hyperplan séparateur, ce qui favorise les données bien classées et situées loin de l'hyperplan. Un exemple de résultat est présenté à la Figure 3.3(c). Ce critère ne garantit toutefois pas que toutes les données sont bien classées. Dans le cas de données non linéairement séparables, cette méthode assure également une convergence vers la configuration minimisant l'erreur quadratique, tel que montré à la Figure 3.3(d).

Les exemples de la Figure 3.3 soulèvent deux problèmes. Le premier est qu'un discriminant linéaire est bien adapté uniquement si les données sont linéairement séparables. La section suivante présente les notions de fonctions de base et de noyaux, qui sont fréquemment utilisés en apprentissage pour des problèmes non linéairement séparables. Le deuxième problème est que, pour maximiser les chances de bien généraliser les données, l'hyperplan séparateur

1. Voir `http://en.wikipedia.org/wiki/Perceptron`.



(a) Perceptron.

(b) Perceptron.

(c) Moindres carrés.

(d) Moindres carrés.

FIGURE 3.3: Exemples de discriminants linéaires obtenus en utilisant les méthodes du perceptron et des moindres carrés.

obtenu devrait être positionné de sorte que toutes les données soient situées le plus loin possible de celui-ci. Par exemple, la solution de la Figure 3.3(c) est préférable à celle de la Figure 3.3(b). La section 3.2.3 présente un algorithme permettant d'accomplir cette tâche.

3.2.2 Fonctions de base et noyaux

Pour des données non linéairement séparables, une solution consiste à projeter les données avec une *fonction de base* dans un espace à plus haute dimensionnalité. Cette fonction, $\boldsymbol{\phi} = [\phi_1 ... \phi_K] : \mathbb{R}^D \mapsto \mathbb{R}^K$, est généralement une transformation non-linéaire. Le classificateur (discriminant linéaire) s'exprime alors par :

$$h(\mathbf{x}|\mathbf{w}, w_0) = \sum_{j=1}^{K} w_j \phi_j(x_j) + w_0 \tag{3.9}$$

Le choix judicieux de la fonction de base peut rendre les données linéairement séparables dans le nouvel espace $\mathbb{R}^K$. Des exemples de fonctions de base sont :

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_1 x_2 \end{bmatrix} \tag{3.10}$$

$$\phi_j(\mathbf{x}) = x_1^{j-1} \tag{3.11}$$

Cette notion amène à introduire le concept de noyau qui, pour deux données $\mathbf{x}$ et $\mathbf{y}$, est défini par $K : X \times X \to \mathbb{R}$ tel que :

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y}) \tag{3.12}$$

Un noyau peut être interprété comme une mesure de similarité entre deux données. Pour son utilisation en apprentissage automatique, un noyau doit respecter la condition de Mercer [2]. Des exemples de noyaux populaires sont le noyau polynomial et le noyau gaussien (RBF), respectivement définis comme suit :

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d, \quad c > 0 \tag{3.13}$$

$$K(\mathbf{x}, \mathbf{y}) = \exp\left[-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right], \quad \sigma > 0 \tag{3.14}$$

Pour le noyau polynomial (3.13), un exemple de projection obtenue, pour des données en deux dimensions et $d = 2$, est :

$$K(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \\ \sqrt{2c}x_1 \\ \sqrt{2c}x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1 y_2 \\ \sqrt{2c}y_1 \\ \sqrt{2c}y_2 \\ c \end{bmatrix} \tag{3.15}$$

Dans le cas du noyau RBF (3.14), il est intéressant de remarquer que $\boldsymbol{\phi}(x)$ possède un nombre infini de dimensions. En pratique, il est possible et souvent plus efficace de calculer directement la valeur de K, sans passer par le calcul de $\boldsymbol{\phi}$ et du produit scalaire. Certains algorithmes d'apprentissage, comme le *Support Vector Machine* présenté à la section suivante, n'utilisent d'ailleurs que cette valeur de K. Cette possibilité démontre bien la puissance de l'utilisation des noyaux.

3.2.3 Support Vector Machine

Tel que mentionné à la section 3.2.1, pour des données linéairement séparables et un discriminant linéaire, l'hyperplan séparateur optimal est celui qui maximise la distance des données de chaque classe par rapport à celui-ci. En effet, un tel positionnement de l'hyperplan maximise la probabilité qu'une nouvelle donnée tombe du bon côté de cette frontière de décision.

2. Voir `http://fr.wikipedia.org/wiki/Théorème_de_Mercer`

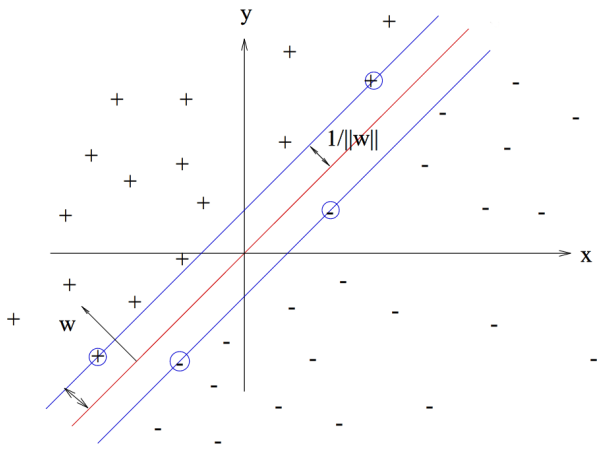


FIGURE 3.4: Maximisation de la marge géométrique avec SVM. Les lignes bleues représentent la marge géométrique, de chaque côté de l'hyperplan séparateur (en rouge). Les données qui sont encerclées sont des vecteurs de support. Image tirée de : `http://fr.wikipedia.org/wiki/Machine_à_vecteurs_de_support`.

La distance entre la frontière de décision et les données les plus proches est appelée la *marge géométrique*.

L'algorithme *Support Vector Machine* (SVM) [4] résout le problème de classification en maximisant la marge géométrique. La Figure 3.4 présente un exemple de résultat désiré. Les données les plus proches de l'hyperplan (qui délimitent la marge) sont appelées les *vecteurs de support* et sont encerclées sur la figure. Le problème d'optimisation se présente sous la forme suivante :

$$\begin{aligned} \min_{\mathbf{w},w_0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{sujet à} \quad & y_t\left(\mathbf{w}^T\mathbf{x}_t + w_0\right) \geq 1, \quad \forall t \end{aligned} \tag{3.16}$$

où $(\mathbf{x}_t, y_t)$ sont les exemples de l'ensemble d'entraînement. La largeur de la marge géométrique est donnée par $1/\|\mathbf{w}\|$. La minimisation du terme $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ maximise donc la marge, alors que la contrainte $y_t\left(\mathbf{w}^T\mathbf{x}_t + w_0\right) \geq 1$ assure que les données sont classées du bon côté de l'hyperplan.

Le problème tel que défini à l'équation (3.16) ne tolère aucune erreur de classification par l'hyperplan. Pour permettre la classification de données non linéairement séparables, le problème peut être redéfini en introduisant des variables d'erreur, ξ_t, permettant aux données de se situer à l'intérieur de la marge ou du mauvais côté de l'hyperplan [10]. La valeur de ces variables est $\xi_t = 0$ pour les données bien classées et situées à l'extérieur de la marge, $0 < \xi_t < 1$ pour les données bien classées mais situées à l'intérieur de la marge et $\xi_t > 1$ pour les données mal classées. Cette nouvelle formulation du SVM est dite à *marges souples*. Le

problème d'optimisation correspondant est le suivant :

$$\begin{aligned} \min_{\mathbf{w},w_0,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{t=1}^{N}\xi_t, \quad C > 0 \\ \text{sujet à} \quad & y_t\left(\mathbf{w}^T\phi(\mathbf{x}_t) + w_0\right) \geq 1 - \xi_t, \quad \forall t \\ & \xi_t \geq 0. \end{aligned} \tag{3.17}$$

où N est le nombre d'exemples de l'ensemble d'entraînement. Le terme $\sum_{i=t}^{N}\xi_t$ est l'erreur associée aux données non linéairement séparables. La variable C est un hyperparamètre de régularisation qui permet de contrôler la pénalité accordée aux erreurs, offrant ainsi un compromis entre le nombre d'erreurs commises et la largeur de la marge.

Le problème d'optimisation du SVM peut être reformulé à nouveau pour utiliser une fonction noyau [3]. Pour ce faire, il faut convertir le problème en un problème de maximisation (formulation duale). Cette formulation, couramment utilisée, a l'avantage d'être toujours résoluble par programmation quadratique. Sa complexité algorithmique est de $O(N^3)$ en temps et $O(N^2)$ en mémoire.

Le SVM fait partie des algorithmes les plus performants à ce jour. Il est cependant peu adapté aux très grands jeux de données, étant donnée sa complexité en fonction du nombre d'exemples (dans la formulation duale).

3.2.4 Bootstrapping

Le *bootstrapping* [12] consiste à échantillonner aléatoirement et avec remise $n \leq N$ données dans un ensemble de départ contenant N données. L'échantillonnage s'effectuant avec remise, il est probable que certaines données soient tirées plus d'une fois. Pour le cas particulier où $n = N$, la probabilité qu'une donnée particulière ne soit jamais tirée lors les n piges est :

$$\left(1 - \frac{1}{N}\right)^n \approx \frac{1}{e} = 0.368 \tag{3.18}$$

On peut donc s'attendre à ce qu'en moyenne $1 - 0.368 = 63.2\%$ des données originales soient présentes dans l'ensemble de données généré.

3.2.5 Bagging

Le *bagging* [5], ou *bootstrap aggregating*, est une technique d'apprentissage qui consiste à entraîner, pour un même problème, plusieurs prédicteurs en fonction d'ensembles d'exemples légèrement différents. Ces ensembles d'exemples d'entraînement sont générés par bootstrapping à partir d'un ensemble d'exemples initial. Les prédicteurs obtenus sont par conséquent

3. Détails sur `http://fr.wikipedia.org/wiki/Machine_à_vecteurs_de_support`.

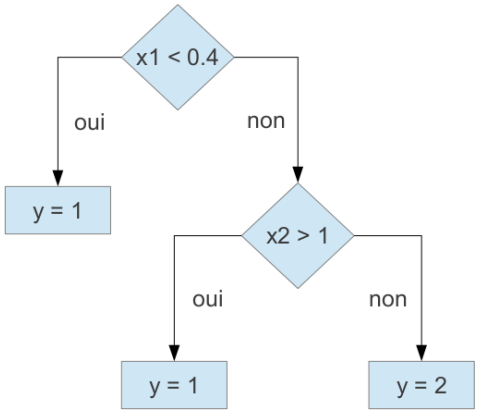


FIGURE 3.5: Arbre de décision simple.

susceptibles de varier légèrement d'un à l'autre. Pour prédire la classe d'une donnée, les sorties de ces prédicteurs sont combinées, par exemple par un vote de majorité. Cette technique est efficace pour augmenter la stabilité et la précision d'un algorithme d'apprentissage. Elle permet également d'éviter le sur-apprentissage.

3.2.6 Random forest

L'algorithme d'apprentissage *random forest* [6] génère un modèle de classification composé de plusieurs arbres de décisions. Un arbre de décision est un arbre dans lequel chacun des noeuds agit comme un classificateur simple, tel qu'un discriminant linéaire, qui sépare les données selon certaines caractéristiques. Les feuilles de l'arbre correspondent à des régions spécifiques de l'espace des caractéristiques, auxquelles une étiquette est attribuée. Un exemple simple d'arbre de décision est présenté à la Figure 3.5.

Dans random forest, les arbres de décision sont générés tels que, à chaque noeud d'un arbre, uniquement un sous-ensemble de caractéristiques est considéré. Ceci permet de réduire significativement la complexité du modèle de classification obtenu. Les sous-ensembles de caractéristiques sont échantillonnés aléatoirement pour chaque noeud dans le but de générer des modèles de discrimination stochastiques. Les arbres sont également entraînés sur des ensembles de données différents obtenus par bootstrapping. Ces conditions d'entraînement font en sorte que les arbres de décision sont généralement différents les uns des autres, situation pour laquelle le bagging s'avère particulièrement efficace. Pour effectuer des prédictions sur de nouvelles données, l'algorithme exploite donc cette idée et combine, par vote de majorité ou autre, les prédictions obtenues pour chacun des arbres. Cette stratégie de combiner par bagging des prédicteurs stochastiques permet d'éliminer naturellement une partie du bruit présent dans les données et de limiter le sur-apprentissage.

Random forest a l'avantage d'être rapide et de pouvoir s'étendre efficacement à des problèmes de classification sur des ensembles de données volumineux et à haute dimensionnalité. Par

ailleurs, même si cet algorithme produit un modèle de classification discriminatif (sans l'estimation des densités de probabilité), il est possible d'obtenir une estimation de la probabilité qu'une donnée appartienne à une certaine classe. Une méthode simple pour ce faire est de calculer la proportion des arbres de décision en accord avec la prédiction d'une classe. Soient $h_k(\mathbf{x}|\boldsymbol{\theta})$ le modèle de prédiction d'un arbre k et n le nombre total d'arbres combinés. La probabilité qu'une donnée $\mathbf{x}$ appartienne à une classe C_i est alors :

$$P(C_i|\mathbf{x}) \approx \frac{\sum_{k=1}^{n} h_k(\mathbf{x}|\boldsymbol{\theta})}{n} \tag{3.19}$$

Chapitre 4

Calage en environnements naturels

Alors que le problème de calage est depuis quelques années bien maîtrisé en environnements intérieurs ou extérieurs structurés, celui-ci présente encore des difficultés pour les environnements naturels (non structurés) comme par exemple les forêts. La grande hétérogénéité des surfaces présentes dans ce type d'environnements rend encore plus nécessaire la réalisation d'un calage grossier préalable à l'alignement fin des nuages de points.

Dans le cas général, une solution simple et sans doute la plus rapide pour réaliser le calage grossier est d'estimer le déplacement effectué par le robot entre les deux scans. Cette estimation peut se faire en utilisant les mesures de capteurs proprioceptifs. Ces dernières risquent toutefois, dans certaines situations, de ne pas être suffisamment fiables. Pour l'odométrie par exemple, un terrain glissant ou très cahoteux, d'ailleurs souvent rencontré en forêt, engendre des mesures erronées. Or, la robustesse et le temps de calcul nécessaire au calage final dépendent en grande partie de la qualité du calage grossier initial. Si les paramètres $\mathbf{R}$ et $\mathbf{t}$ du calage grossier ne tombent pas dans le bassin de convergence du minimum global de la fonction d'erreur de l'alignement fin, alors il est probable que le calage final soit imprécis. Par conséquent, un calage grossier basé sur les mesures de tels capteurs est peu adapté à ce type d'environnements.

Une approche plus adaptée consiste par conséquent à calculer le calage grossier à partir des nuages de points eux-mêmes, tel que présenté à la section 2.3.2. Pour ce faire, l'une des méthodes les plus populaires et performantes à ce jour est le calcul de descripteurs et l'utilisation de l'algorithme *Sample Consensus Initial Alignment* (SAC-IA). À titre de rappel, cet algorithme pige aléatoirement des points dans le premier scan et les associe chacun aléatoirement dans une liste des points les plus potentiellement correspondants (descripteurs similaires) dans le second scan. Les correspondances ainsi établies à chaque itération sont utilisées pour calculer des valeurs potentielles de $\mathbf{R}$ et $\mathbf{t}$. La précision de l'alignement grossier obtenu dépend donc fortement de la qualité des descripteurs. En effet, tandis que des descripteurs robustes et discriminatifs engendrent une majorité de correspondances valides, des descripteurs instables

ou trop similaires les uns des autres engendrent une proportion importante de correspondances erronées. Par conséquent, pour maximiser les chances de réussite de l'algorithme, il est primordial que les descripteurs utilisés soient robustes et qu'ils se distinguent dans l'environnement.

Comme il sera présenté à la section 4.3, de nombreux descripteurs calculés dans les environnements naturels ne sont pas fiables. La présence de ces mauvais descripteurs réduit la probabilité de SAC-IA de piger des correspondances valides (voir l'équation(2.8)). L'algorithme requiert donc en moyenne un plus grand nombre d'itérations pour trouver un alignement correct (voir l'équation (2.9)), ce qui implique un temps de calcul plus élevé.

Ce chapitre présente une solution que nous proposons [22] pour améliorer le calage en environnements naturels, y compris ceux présentant une végétation dense (non structurés). Cette solution a été développée en collaboration avec Alexandre Drouin et François Laviolette du Groupe de Recherche en Apprentissage Automatique de l'Université Laval (GRAAL).

4.1 Aperçu de la méthode proposée

L'approche proposée consiste à appliquer un algorithme d'apprentissage automatique, sous un paradigme appelé *positive and unlabeled learning* [13], afin de distinguer, pour un certain environnement, les descripteurs fiables (stables et distinctifs) de ceux qui ne le sont pas. Le classificateur entraîné permet de filtrer les descripteurs (de même que les points auxquels ils ont été calculés) afin d'augmenter la proportion de descripteurs fiables. L'objectif est à la fois d'éliminer le plus possible de descripteurs non fiables et de conserver le plus possible de descripteurs fiables. Ceci permet d'améliorer deux aspects du calage réalisé avec l'algorithme SAC-IA :

– Robustesse : une plus grande proportion de descripteurs fiables engendre une plus grande probabilité pour l'algorithme de piger des correspondances valides. La probabilité de succès de SAC-IA, pour un nombre d'itérations N_{iter} donné, est par conséquent plus élevée.
– Temps de calcul : en réduisant le nombre de points considérés par l'algorithme (ceux correspondants aux descripteurs classés fiables), le calcul d'erreur de l'alignement effectué à chaque itération requiert moins de temps. Chaque itération se termine donc plus rapidement.

Également, une meilleure précision de l'estimation de correspondances offre la possibilité de diminuer le nombre d'itérations effectuées. Le temps de calcul requis par SAC-IA peut donc être significativement réduit par le filtrage des descripteurs.

Le diagramme de blocs de la solution proposée est présenté à la Figure 4.1. L'une des forces de la méthode est que le robot apprend par lui-même, sans aucune intervention humaine nécessaire. Les données d'entraînement sont générées et étiquetées automatiquement à partir des deux premiers scans. L'apprentissage peut donc être réalisé à la volée, pour un type

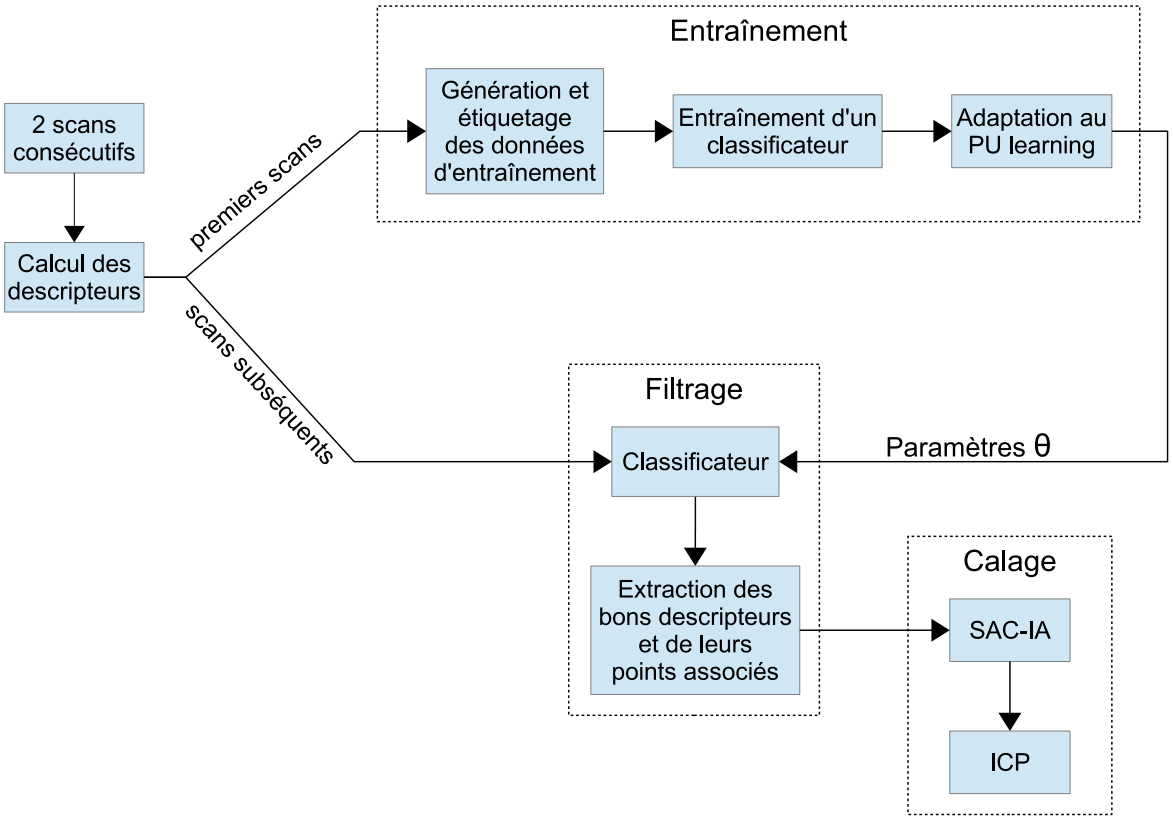


FIGURE 4.1: Diagramme de blocs de la solution proposée pour améliorer le calage en environnements naturels. Les paramètres $\boldsymbol{\theta}$ sont estimés à partir d'un ensemble d'entraînement étiqueté par le robot lui-même. Les descripteurs qui sont classés comme étant fiables par le classificateur entraîné, de même que les points auxquels ils ont été calculés, sont extraits des scans et utilisés par la suite pour le calage.

d'environnement donné. Dans une application réelle, cet avantage permettrait au robot de s'adapter naturellement aux changements de l'environnement au fur et à mesure qu'il se déplace, soit en se réentraînant périodiquement, soit lorsque nécessaire.

La section 4.2 présente d'abord les travaux connexes publiés dans la littérature. La section 4.3 démontre les difficultés associées avec l'utilisation des descripteurs dans les environnements naturels. La méthode proposée est ensuite détaillée à la section 4.4. Les expérimentations et résultats sont présentés au chapitre 5.

4.2 Travaux connexes

Tel qu'explicité à la section 2.4, de nombreux types de descripteurs ont été présentés dans la littérature. Pour s'assurer de la stabilité des descripteurs calculés, certaines méthodes incorporent une étape d'extraction de points d'intérêts. Les SIFT 3D (section 2.4.3) en font partie, mais nécessitent une valeur associée à chaque point, en plus des coordonnées (x, y, z),

pour leur calcul. Cette valeur peut être, par exemple, l'intensité de retour du laser, mais cette information n'est pas toujours disponible et fiable [1]. Le descripteur *Normal Aligned Radial Feature* (NARF) [43] extrait également des points d'intérêts stables avant le calcul des descripteurs, mais ne s'applique qu'aux images de profondeur (telles que fournies naturellement par un capteur 3D comme Kinect) et ne peut se calculer directement sur un nuage de points. Bien qu'il soit possible de convertir un nuage de points en une image de profondeur, cette opération peut altérer l'information initiale.

La suite de ce chapitre s'attarde uniquement aux descripteurs FPFH et SHOT, étant donné qu'ils sont parmi les plus performants actuellement et directement applicables aux nuages de points acquis de scanner lasers. Comme il sera démontré à la section suivante, leurs performances sont toutefois limitées dans les environnements naturels non structurés. Par ailleurs, avec un objectif similaire à celui de la méthode proposée dans ce chapitre, Rusu *et al.* [34] ont suggéré une méthode pour filtrer les FPFH, de sorte que seuls les plus utiles sont conservés pour établir des correspondances et aligner les nuages de points. Leur méthode est cependant peu adaptée aux environnements naturels, ce qui sera démontré à la section 4.3.1.

Il vaut la peine de mentionner qu'une méthode de calage basée sur l'extraction de troncs d'arbres a récemment été proposée pour des nuages de points acquis en environnements forestiers [42]. Bien qu'elle soit performante, cette méthode assume que des troncs approximativement parallèles sont présents dans l'environnement, visibles et bien échantillonnés par le scanner laser. Cette condition n'est pas toujours respectée : par exemple, les troncs peuvent être obstrués par du feuillage, trop petits (pas capturés par l'échantillonnage du laser) ou tordus (donc non parallèles), ou encore l'environnement peut être constitué uniquement de petit arbustes ou de buissons.

Le concept d'un robot mobile collectant des données de capteurs pour entraîner un système de navigation autonome a déjà été étudié. Par exemple, des méthodes ont été présentées pour apprendre les modèles d'apparence de surfaces navigables dans le désert [11], pour apprendre des modèles d'interaction robot-terrain par expérimentations [21] et pour faire de la prédiction de dérapage en utilisant un apprentissage par expérience [1]. La méthode proposée dans ce chapitre est similaire par le fait que le robot s'entraîne de lui-même, sans intervention humaine requise, en se basant sur des données qu'il a acquises dans l'environnement. Par ailleurs, l'apprentissage automatique a également été démontré comme étant applicable aux descripteurs. Sim et Dudek [41] ont présenté une méthode pour évaluer et apprendre, dans un environnement donné, un ensemble de points de repère utiles pour l'estimation de la pose d'un robot. Grabner *et al.* [18] ont présenté une méthode pour apprendre, à la volée, les descripteurs de points d'intérêts appartenant à un objet afin d'en améliorer le suivi dans une image vidéo. L'apprentissage automatique a aussi été utilisé pour trouver de nouvelles

1. L'intensité de retour du laser varie grandement en fonction du type, de la couleur et de la distance et de l'orientation de la surface sur laquelle le rayon est réfléchi.

représentations de descripteurs (c.-à-d. types de descripteurs) pour les images, en optimisant les paramètres de la configuration DAISY [2] [49], puis en utilisant un ensemble plus complexe de blocs prédéfinis [7]. Dans les deux cas, les paramètres des blocs ont été optimisés pour maximiser le pouvoir discriminatif du descripteur résultant. Suivant la même idée, Trzcinski *et al.* [48] ont appliqué le *boosting* [3] pour trouver une projection non-linéaire optimale de sections d'images vers un espace de caractéristiques. Leurs expérimentations ont mené à des descripteurs plus performants que le SIFT.

4.3 Descripteurs en environnements naturels

Les environnements naturels se définissent souvent par la présence de feuillage, d'herbes ou de branchages. La surface de ces objets a la particularité d'être très irrégulière et constituée de petits fragments (ex : feuilles). Les scanners laser ayant une résolution angulaire limitée, les fortes variations du profil de ces surfaces peuvent ne pas être mesurées adéquatement. Pour une résolution angulaire typique de 0.2° par exemple, la distance entre les mesures d'une surface située à 30 m du capteur est d'un peu plus de 10 cm. On peut considérer que cette distance correspond environ aux dimensions moyennes d'une feuille d'arbre. Il est donc impossible, à cette distance, d'échantillonner plus d'un point par feuille. De plus, les feuilles plus petites ou dont l'angle d'incidence par rapport au faisceau laser est inférieur à 90° ne sont pas garanties d'être capturées. Le feuillage (et le branchage) peut donc, dans certaines conditions, ne pas être échantillonné suffisamment densément pour que les mesures acquises permettent de bien représenter sa surface.

La Figure 4.2 montre le résultat d'une de nos simulations reproduisant ces conditions pour un feuillage [4] situé à 30 m. Pour faciliter la visualisation, la simulation a été effectuée pour une tranche de feuillage en deux dimensions. On observe qu'à cette distance, il n'y a en effet jamais plus d'un point mesuré par feuille et que certaines feuilles ne sont pas observées. Pour une surface ainsi sous-échantillonnée, il est clair que l'estimation du profil de la surface n'est pas fiable : un changement de point de vue du capteur peut entraîner des mesures significativement différentes. Il faut noter que pour des herbes, de petits arbustes ou de petites branches, la dimension des fragments composant la surface est plus petite, faisant alors survenir ce problème à plus courte distance. Le sous-échantillonnage du feuillage peut donc causer problème lors de traitements effectués sur les nuages de points.

Un autre problème rencontré avec le feuillage est la variation de la forme observable du profil de la surface en fonction de l'angle du point de vue. Les feuilles, orientées dans tous

2. Représentation générique du calcul d'un descripteur par un ensemble de blocs fonctionnels. Chaque blocs réalise une opération spécifique (filtrage, dérivée, etc.) et est paramétrable.

3. Technique consistant à combiner des classificateurs faibles (au moins aussi performant que le hasard), qui ne sera pas détaillée ici. Voir `http://en.wikipedia.org/wiki/Boosting_(machine_learning)` pour des détails.

4. Pour des fins de simplicité, un feuillage carré a été simulé. Les feuilles sont des plans de 10×10 cm, dont la position et l'orientation varient aléatoirement.

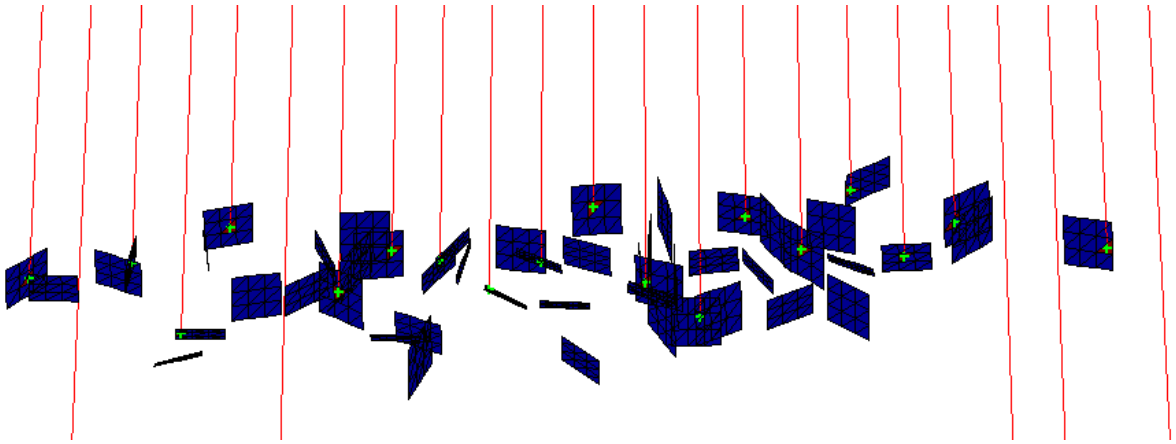
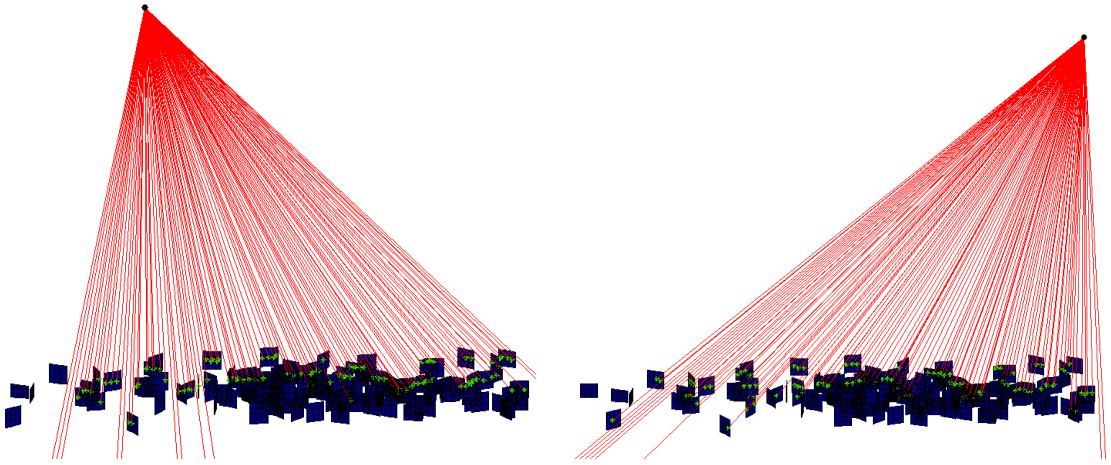


FIGURE 4.2: Simulation de l'échantillonnage d'un feuillage (carré) par un scanner laser 2D (résolution angulaire de 0.2°) situé à une distance de 30 m. Les points mesurés par le laser sont en vert.

les sens, peuvent en effet faire obstruction les unes envers les autres lors de changements de points de vue. Cela peut entraîner des variations de la forme apparente du profil de la surface mesurée, même lorsque l'échantillonnage est bon. Ce problème peut donc survenir autant aux courtes qu'aux grandes distances. La Figure 4.3 compare les points mesurés lors d'une de nos simulations d'échantillonnage d'un feuillage situé à 6 m, pour deux positions d'un scanner laser (résolution de 0.2°). Ces positions, A et B, sont séparées de 2 m l'une de l'autre. Pour un tel déplacement, l'angle du point de vue sur le feuillage varie d'environ 19° entre les positions. Comme le montre le graphique de la Figure 4.3(c), les deux nuages de points obtenus présentent des différences significatives.

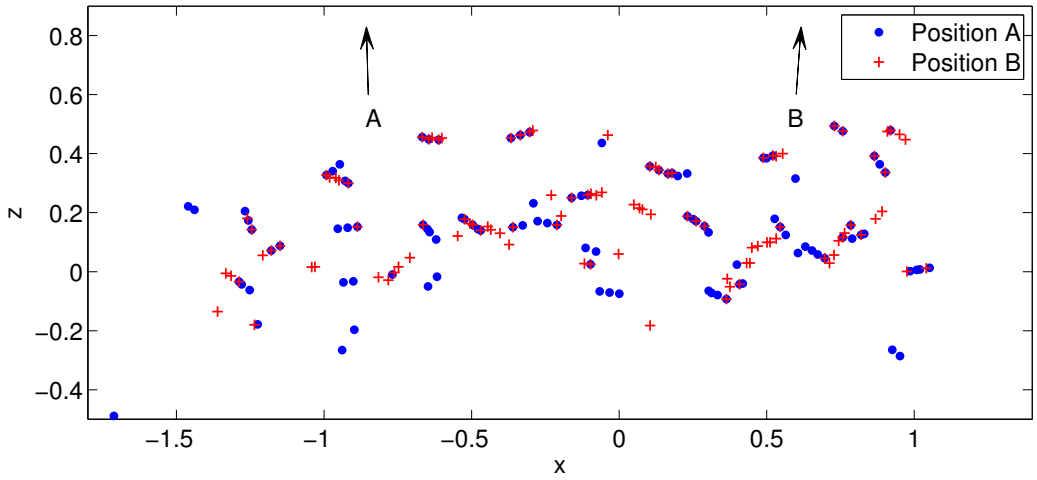
La variation des points mesurés dans le feuillage entre les scans, qu'elle soit causée par un échantillonnage inadéquat ou un changement de point de vue, implique une variation des descripteurs calculés pour ces points. Des descripteurs de points situés au même endroit dans l'environnement peuvent alors posséder des valeurs différentes entre les scans. Ces différences peuvent être suffisamment significatives pour engendrer un grand nombre de correspondances invalides. Par conséquent, on peut supposer que les descripteurs de points mesurés dans le feuillage sont fondamentalement instables, donc non fiables pour l'estimation de correspondances et le calage.

Pour vérifier cette affirmation, des scans d'un même feuillage ont été simulés pour différents points de vue du capteur (résolution de 0.2°), en faisant varier l'angle d'observation θ_{PDV} sur le feuillage. Les points de vue ont été choisis tels qu'ils soient toujours orientés vers le centre C du feuillage et que leur distance d_{PDV} par rapport à C soit constante. Les différentes positions du capteur décrivent donc un arc de cercle autour du feuillage, tel qu'illustré à la Figure 4.4. Le feuillage a été généré tel qu'il soit entièrement couvert par le champ de vision du scanner laser pour chacun des points de vue. Les descripteurs FPFH et SHOT ont ensuite été calculés



(a) Position A.

(b) Position B.



(c) Comparaison des coordonnées, dans le repère global, des points mesurés pour les positions A et B du scanner laser.

FIGURE 4.3: Simulations de l'échantillonnage d'un feuillage (carré) situé à 6 m, pour deux positions d'un scanner laser (résolution de 0.2°) séparées de 2 m. Les figures 4.3(a) et 4.3(b) montrent les points de vue du scanner laser, alors que la figure 4.3(c) compare les coordonnées des points mesurés, dans le repère global, pour les deux scans.

dans chacun des scans simulés. Des tests d'estimation de correspondances ont été effectués entre les scans, pour ces deux types de descripteurs, afin de mesurer l'impact du changement de point de vue sur la précision des correspondances établies. Plus précisément, pour chacun des points de vue, les correspondances ont été établies avec le scan acquis à $\theta_{PDV} = 0°$. Pour ces tests, les correspondances ont été établies systématiquement avec le plus proche voisin dans l'espace des descripteurs, sans tenir compte d'un seuil t_{corr}. Une distance euclidienne maximale, fixée à 7 cm [5], a été utilisée pour valider les correspondances. Ces simulations de

5. Cette distance est un peu plus petite que la dimension des feuilles simulées (10×10 cm) et un peu plus

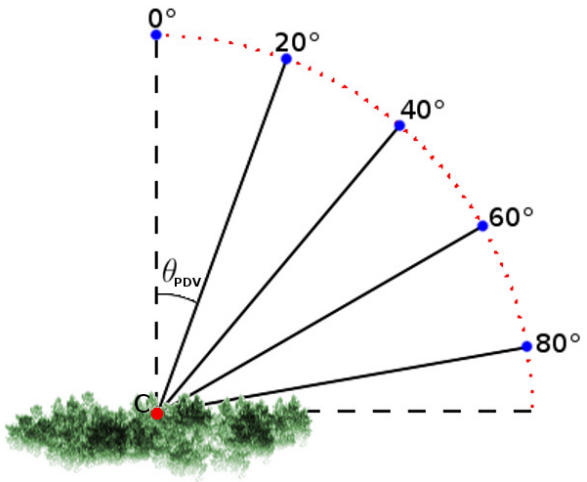


FIGURE 4.4: Configuration des points de vue (en bleu) du scanner laser pour simuler l'impact du changement de point de vue sur l'estimation de correspondances dans du feuillage. L'arc de cercle en rouge pointillé montre que les points de vue sont équidistants du point C au centre du feuillage.

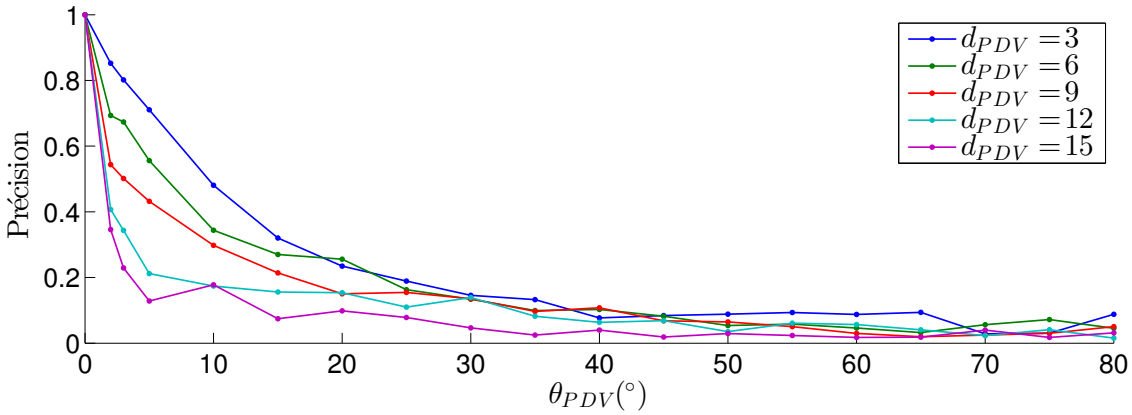
scans et tests d'estimation de correspondances ont été répétés pour différentes valeurs de d_{PDV} afin d'observer la variation des résultats en fonction de la qualité de l'échantillonnage. La Figure 4.5 montre les résultats obtenus pour les deux types de descripteurs testés. D'après ces résultats, il est clair que la précision des correspondances établies dans le feuillage diminue en fonction de la distance d_{PDV} du scanner laser (qualité de l'échantillonnage) et qu'elle décroît rapidement en fonction de l'angle θ_{PDV} entre les points de vue.

En comparaison, la Figure 4.6 présente les résultats d'une même simulation effectuée pour une surface plus lisse. La précision des correspondances établies dans ce cas diminue beaucoup plus lentement en fonction de l'angle θ_{PDV}, surtout pour les descripteurs FPFH, et varie plus graduellement en fonction de la distance d_{PDV} du scanner laser.

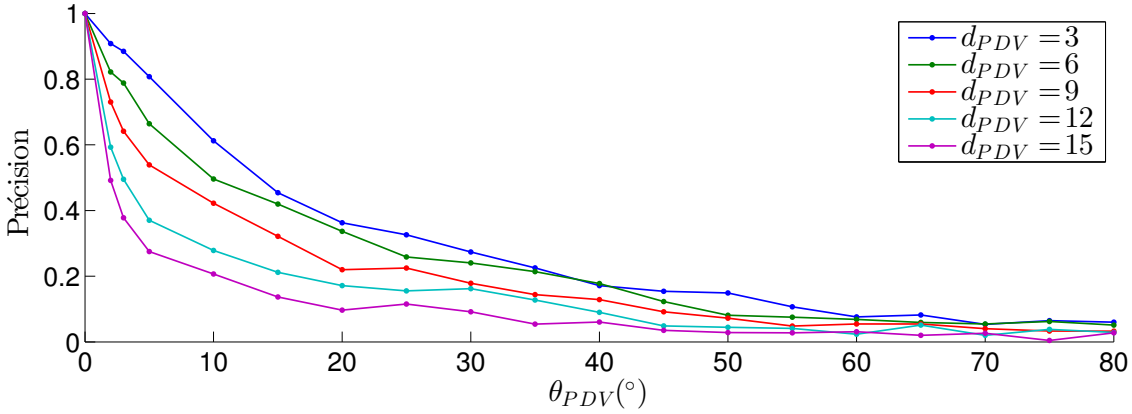
Ces simulations montrent que, même en utilisant des types de descripteurs parmi les plus performants, les points mesurés dans le feuillage sont peu fiables pour estimer des correspondances entre les scans. Il est d'ailleurs important de spécifier que ces simulations supposent que les feuilles sont statiques. En réalité, les feuilles (et les branches) peuvent bouger entre les scans à cause du vent, ce qui peut dégrader davantage les performances. Également, aucun bruit n'a été simulé sur les mesures du scanner, même si en pratique ce bruit est inévitable.

Enfin, des tests d'estimation de correspondances ont été effectués sur des scans réels acquis en environnements naturels, provenant de l'ensemble de données *Hannover* [50]. Cet ensemble de données a été acquis sur un campus universitaire, en présence de bâtiments, mais aussi

grande que la distance entre les mesures du scanner laser (5 cm) lorsque $d_{PDV} = 15$ m (distance maximale du capteur testée pour cette simulation).



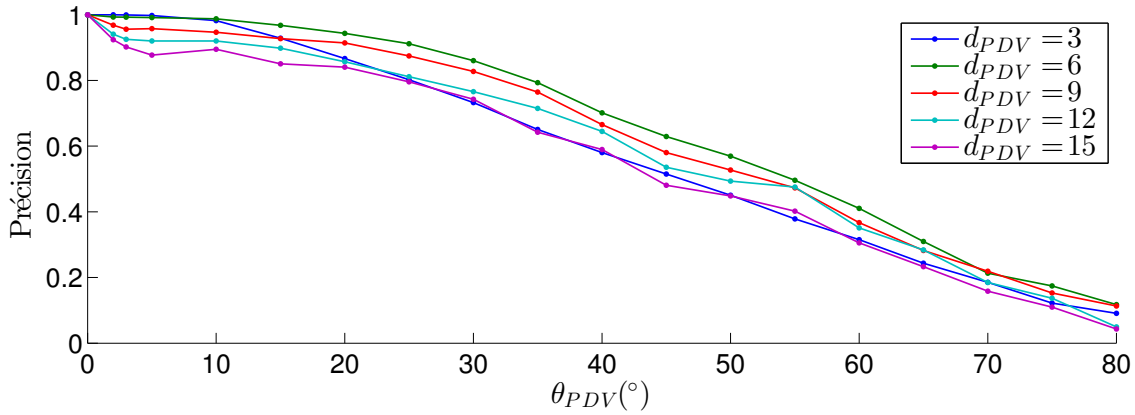
(a) FPFH.



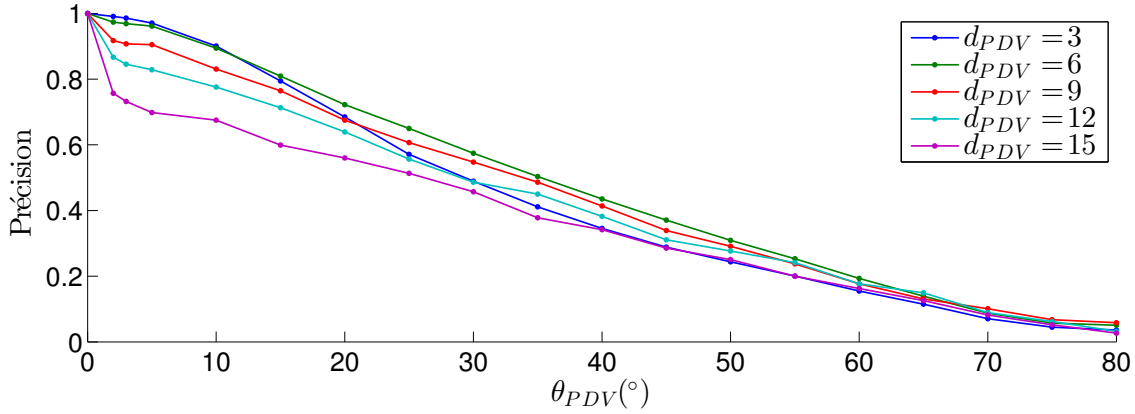
(b) SHOT.

FIGURE 4.5: Précision des correspondances établies dans un feuillage simulé en fonction de l'angle entre les points de vue des scans, pour les descripteurs FPFH et SHOT.

de buissons, d'arbres et d'arbustes. La Figure 4.7 montre des exemples de résultats obtenus pour deux scans consécutifs (22-23), en utilisant les descripteurs FPFH et SHOT. Ces scans présentent en particulier trois arbres et un coin de bâtiment. Le déplacement du robot entre ces scans est très petit : la translation est de 14.5 cm et la rotation est négligeable. Dans ce cas, les scans testés étant réels, les correspondances ont été validées selon les différences d'angle et de distance par rapport au capteur des points correspondants, en utilisant la méthode qui sera présentée à la section 4.4.1. Cette méthode de validation permet de tenir compte du modèle de bruit du scanner laser, de même que du fait que la densité des points mesurés n'est pas constante en fonction de leur distance par rapport au scanner laser. Pour les deux types de descripteurs, le seuil t_{corr} pour l'estimation de correspondances a été fixé à la valeur correspondant au coude de la courbe de précision et rappel. En utilisant les FPFH, la précision des correspondances établies est de $1798/9078 \approx 20\%$ et le rappel de $1798/12355 \approx 14.5\%$. En



(a) FPFH.

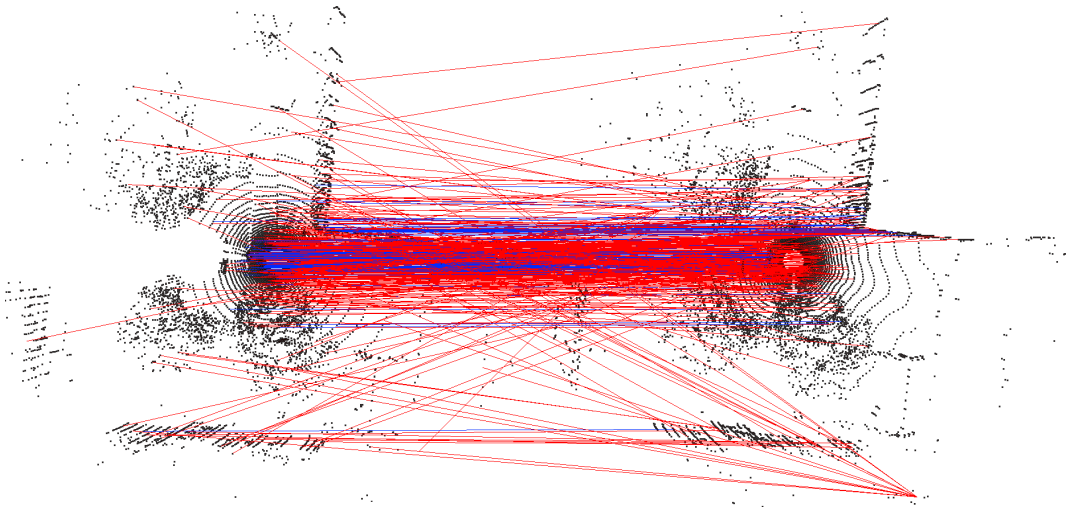


(b) SHOT.

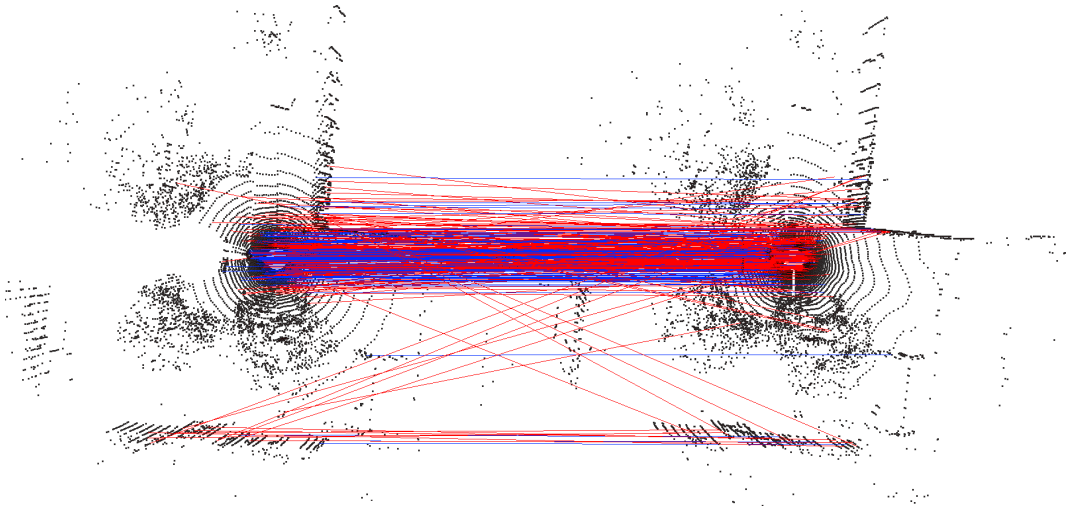
FIGURE 4.6: Précision des correspondances établies pour une surface plus lisse (simulée) en fonction de l'angle entre les points de vue des scans, pour les descripteurs FPFH et SHOT.

utilisant les SHOT, la précision est de $1868/7340 \approx 25\%$ et le rappel de $1868/12355 \approx 15\%$. Il est important de remarquer que très peu de correspondances ont été établies dans le feuillage, indiquant que peu de descripteurs similaires existent entre les scans dans ces parties de l'environnement. La majorité des correspondances valides ont été établies près du coin de mur et au centre, où deux humains se tiennent debout (non visible sur l'image présentée). Parmi les quelques correspondances qui ont été établies dans le feuillage, la majorité est erronée.

L'ensemble des expérimentations effectuées démontre que les descripteurs des points mesurés dans le feuillage sont instables et, par conséquent, peu fiables pour établir des correspondances entre les nuages de points. Leur présence dans l'ensemble des descripteurs utilisés par SAC-IA réduit donc la probabilité pour l'algorithme de piger des correspondances valides (équation (2.8)) : lorsqu'un descripteur situé dans du feuillage est pigé, les expérimentations



(a) FPFH.



(b) SHOT.

FIGURE 4.7: Correspondances établies entre les scans 22 et 23 de *Hannover* en utilisant les descripteurs FPFH et SHOT (rayon de voisinage $r = 1\,\text{m}$). Les lignes bleues et rouges représentent respectivement les correspondances valides et invalides. Pour alléger les images, les correspondances affichées ont été décimées d'un facteur 20.

ci-haut démontrent qu'il est peu probable qu'une correspondance valide soit établie. Par conséquent, le filtrage préalable de ces descripteurs, pour lequel nous proposons une méthode à la section 4.4, pourrait améliorer la précision et la robustesse du calage, de même que son efficacité computationnelle. La section suivante présente d'abord une méthode de filtrage existante, proposée par les auteurs des FPFH, mais qui est peu adaptée aux environnements naturels.

4.3.1 Filtrage des descripteurs communs

Pour les descripteurs PFH et FPFH, une méthode de filtrage est proposée par Rusu *et al.* [33][34] afin de conserver uniquement les descripteurs les plus utiles pour l'estimation de correspondances. Dans leur cas, les descripteurs utiles sont ceux qui sont les plus distinctifs, c'est-à-dire peu communs dans le nuage de points.

Pour un nuage de points $\mathcal{P}$ et son ensemble de descripteurs $\mathcal{F}_\mathcal{P}$, les auteurs suggèrent de calculer le descripteur moyen $\bar{\mathbf{f}}_\mathcal{P}$, puis de calculer la distance de chaque descripteur $\mathbf{f} \in \mathcal{F}_\mathcal{P}$ par rapport à $\bar{\mathbf{f}}_\mathcal{P}$, selon une métrique donnée. Tel que démontré dans [33], la distribution de toutes ces distances peut être approximée par une distribution gaussienne $\mathcal{N}(\mu_d, \sigma_d)$, de moyenne μ_d et d'écart-type σ_d. Les descripteurs pour lesquels la valeur de la distance se situe à l'extérieur de l'intervalle $\mu_d \pm \beta \cdot \sigma_d$, où β est un seuil de filtrage défini par l'utilisateur, peuvent ensuite être sélectionnés en tant que descripteurs moins communs. Ces derniers forment l'ensemble des descripteurs *uniques*, $\mathcal{F}^*_\mathcal{P}$. Selon les auteurs, une valeur de β entre 1 et 2 donne des résultats satisfaisants dans la plupart des cas.

Ce processus peut être répété pour différentes valeurs de rayon r définissant le voisinage lors du calcul des descripteurs. Ceci permet de tenir compte de plusieurs échelles de grandeur physique pour l'identification des descripteurs distinctifs. Les descripteurs marqués uniques pour deux valeurs de rayon r_i consécutives (c.-à-d. pour deux échelles consécutives) sont alors dits *persistants*. Plus formellement, l'ensemble de ces descripteurs persistants, $\mathcal{F}^{**}_\mathcal{P}$, est donné par :

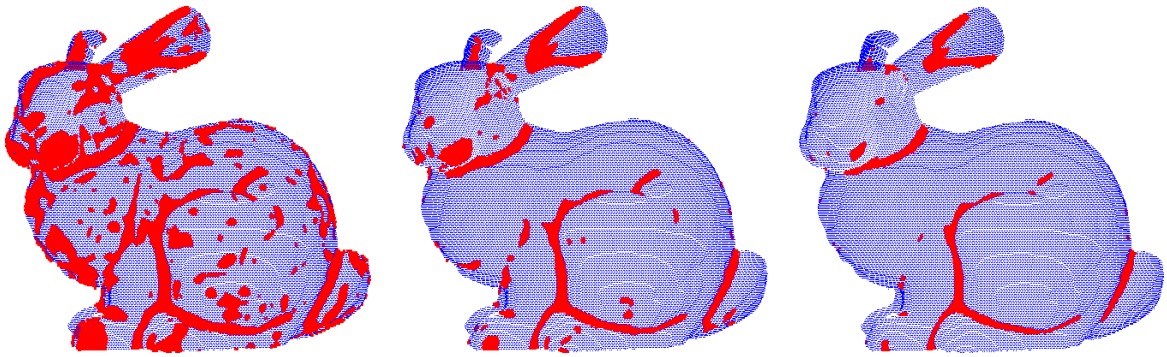
$$\mathcal{F}^{**}_\mathcal{P} = \bigcup_{i=1}^{n-1} \mathcal{F}^*_{\mathcal{P},i} \cap \mathcal{F}^*_{\mathcal{P},i+1} \tag{4.1}$$

où n est le nombre de valeurs de rayons utilisées.

Bien que cette méthode de filtrage assure que les descripteurs conservés correspondent à des points distinctifs dans l'environnement, elle ne garantit pas leur stabilité ou leur fiabilité pour établir des correspondances. En particulier, dans les environnements naturels, les descripteurs bruités dans le feuillage et les branches sont susceptibles de différer significativement d'un à l'autre et par conséquent d'apparaître comme étant distinctifs. Pour illustrer ce problème, la Figure 4.8 compare la localisation des descripteurs FPFH uniques obtenus pour différentes valeurs de β, pour le *Stanford Bunny* (surface relativement lisse et bien échantillonnée) et pour un scan de l'ensemble de données *Hannover*. Alors que le filtrage fait ressortir des points saillants mais stables pour le *Bunny*, les descripteurs conservés sur *Hannover* sont majoritairement situés dans le feuillage.

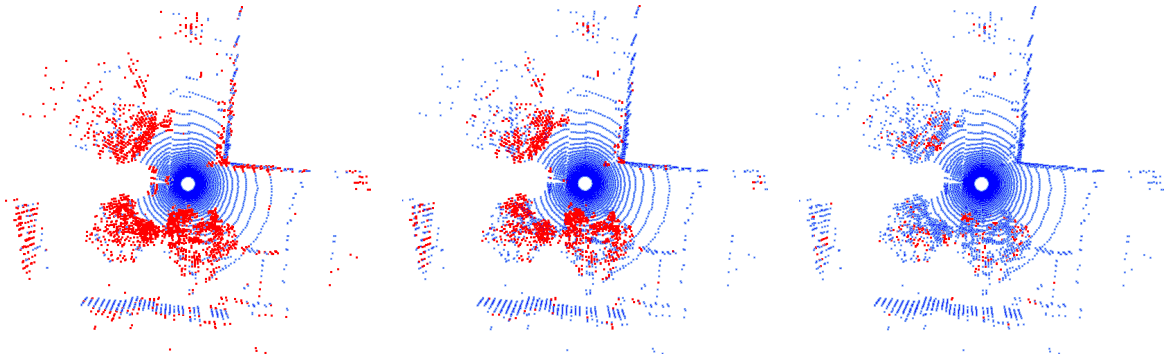
Une analyse plus poussée de l'effet de ce filtrage peut être effectuée en calculant la proportion, parmi tous les descripteurs uniques, de ceux pouvant mener à des correspondances valides [6]. Pour simplifier le texte, ces descripteurs seront appelés *descripteurs valides* dans ce

6. Noter que la valeur de cette proportion est indépendante de la valeur du seuil t_{corr}. Un descripteur peut



$\beta = 1$ $\beta = 1.4$ $\beta = 1.7$

(a) *Stanford Bunny*.



$\beta = 1$ $\beta = 1.4$ $\beta = 1.7$

(b) *Hannover* scan 22.

FIGURE 4.8: Localisation des descripteurs FPFH uniques (en rouge) obtenus par filtrage des descripteurs communs, selon la méthode de Rusu *et al.* [33][34], pour $\beta = \{1, 1.4, 1.8\}$ sur le *Stanford Bunny* ($r = 0.003\,\text{m}$) et un scan de *Hannover* ($r = 1\,\text{m}$).

qui suit. Soient $\mathcal{V}_\mathcal{P}$ et $\mathcal{V}^*_\mathcal{P}$ les ensembles des descripteurs valides, avant et après l'identification des descripteurs uniques respectivement. Un algorithme de filtrage approprié devrait faire augmenter la proportion de descripteurs valides par rapport à celle de départ (avant filtrage), c'est-à-dire que la condition

$$\frac{|\mathcal{V}^*_\mathcal{P}|}{|\mathcal{F}^*_\mathcal{P}|} > \frac{|\mathcal{V}_\mathcal{P}|}{|\mathcal{F}_\mathcal{P}|} \tag{4.2}$$

devrait être respectée. Le graphique de la Figure 4.9 montre les résultats, obtenus par tests d'estimation de correspondances, pour la proportion de descripteurs valides en fonction de β, sur le *Bunny*[7] et sur les scans 22-23 de *Hannover*. Dans le cas du *Bunny*, la proportion de descripteurs valides s'accroît significativement lorsque la valeur de β augmente, indiquant que le filtrage des descripteurs communs par la méthode de Rusu améliore la qualité de l'ensemble

mener à une correspondance valide si son plus proche voisin (dans l'espace des caractéristiques) est réellement correspondant dans l'espace 3D. Aucun seuil t_{corr} n'est donc utilisé pour calculer cette proportion.

7. Du bruit gaussien a été ajouté au scan du *Bunny* pour simuler deux scans différents.

de descripteurs résultant. Dans le cas des scans de *Hannover* cependant, la proportion diminue presque constamment jusqu'à devenir nulle à $\beta = 2$ environ. Cela signifie que, pour ces scans, les descripteurs pouvant mener à des correspondances valides sont filtrés à un taux plus élevé que les autres à mesure que β augmente, ce qui est contraire à l'effet recherché.

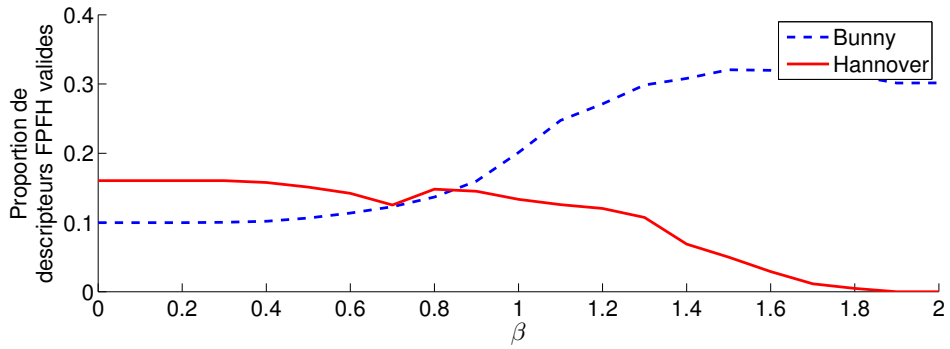


FIGURE 4.9: Effet du paramètre β sur la proportion de FPFH uniques pouvant mener à des correspondances valides sur le *Bunny* et les scans 22-23 de *Hannover*.

Les Figures 4.8 et 4.9 démontrent que, dans les environnements naturels, le filtrage selon la méthode de Rusu peut avoir un impact négatif. En effet, étant donné qu'une grande partie des descripteurs uniques est située dans le feuillage, la proportion de descripteurs fiables après le filtrage peut être inférieure à celle de départ. Même si l'utilisation d'un seuil de correspondances t_{corr} permet de n'établir que les correspondances les plus probables, le nombre réduit de descripteurs valides après le filtrage mène à un plus petit nombre de correspondances valides. Pour illustrer cette conséquence de cette méthode de filtrage, la Figure 4.10 montre le résultat de l'estimation de correspondances entre les scans 22-23 de *Hannover*, en utilisant $\beta = 1.2$. Le seuil t_{corr} a été fixé à la même valeur qu'à la Figure 4.7(a). La précision des correspondances établies est augmentée à $233/714 \approx 33\%$, mais avec un rappel de seulement $233/12355 \approx 1.9\%$. Encore une fois, les correspondances valides ont été établies en majorité sur le coin du bâtiment et les humains au centre, alors que celles établies dans le feuillage sont en majorité erronées. Dans un environnement présentant une végétation dense, on peut supposer que l'absence de structure empirerait les résultats.

Bien qu'elle soit efficace pour les surfaces relativement lisses et bien échantillonnées, la méthode de Rusu pour filtrer les descripteurs communs n'est par conséquent pas bien adaptée aux environnements naturels. La méthode que nous proposons à la section suivante permet non seulement de filtrer les descripteurs instables, mais par le fait même ceux qui ne sont pas assez distinctifs dans l'environnement.

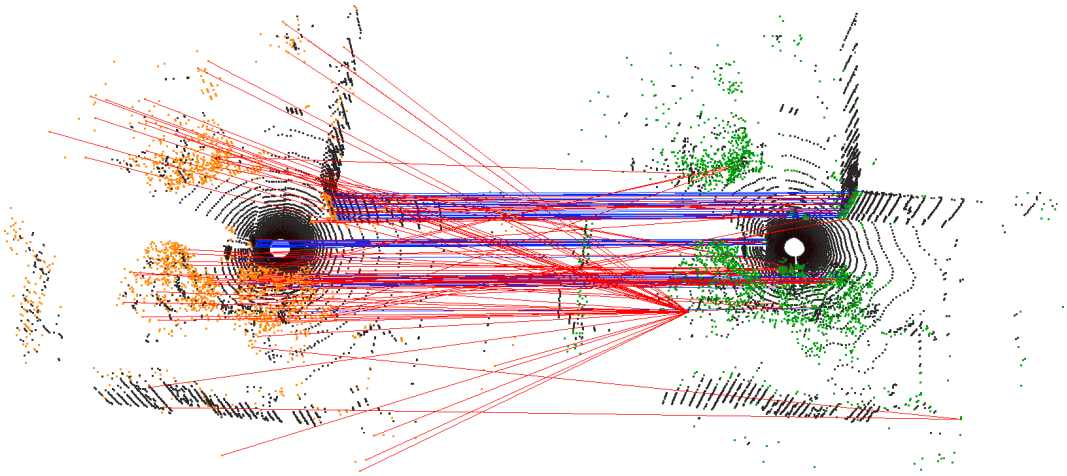


FIGURE 4.10: Correspondances établies entre les scans 22 et 23 de *Hannover* en utilisant les descripteurs FPFH uniques ($r = 1\,\text{m}$), pour $\beta = 1.2$. Les correspondances affichées ont été décimées d'un facteur 5. Les points orange et verts représentent la localisation des descripteurs uniques.

4.4 Méthode proposée

Cette section détaille la méthode proposée pour filtrer les descripteurs non fiables (c.-à-d. instables ou trop communs) dans un certain environnement.

La sous-section 4.4.1 présente l'approche utilisée pour générer les données d'apprentissage. Le paradigme d'apprentissage *positive and unlabeled* est ensuite expliqué à la sous-section 4.4.2, suivi de la description de l'algorithme d'apprentissage proposé à la sous-section 4.4.3. Les détails concernant le filtrage réalisé avec le classificateur entraîné sont présentés à la sous-section 4.4.4.

4.4.1 Génération des données d'entraînement

Dans l'approche proposée, l'ensemble de données d'entraînement est généré par le robot lui-même. Aucune intervention humaine n'est nécessaire pour récolter ou étiqueter les données. Ceci permet au robot de s'entraîner à la volée et donc de s'adapter naturellement aux changements dans l'environnement au fur et à mesure qu'il se déplace.

La génération des données d'entraînement se base sur les constatations détaillées précédemment à la section 4.3. Comme il a été démontré, pour une surface sous-échantillonnée ou dont le profil est très irrégulier, un changement de point de vue, même petit, peut engendrer une proportion importante de descripteurs instables. L'idée est donc d'effectuer un test d'estimation de correspondances entre deux scans $\mathcal{P}$ et $\mathcal{Q}$, acquis par le robot à des points de vue légèrement différents, puis d'évaluer les correspondances établies, en fonction d'un calage de ces scans, afin d'identifier les descripteurs qui sont valides. D'après ce qui a été dit

précédemment, on peut s'attendre à ce que la majorité des descripteurs de points situés dans le feuillage (ou branches, etc.) mène à des correspondances erronées, alors que la majorité de ceux qui sont stables et distinctifs (dans cet environnement) à des correspondances valides.

Il faut réitérer le fait que le déplacement entre les scans $\mathcal{P}$ et $\mathcal{Q}$ est petit. Cela implique, d'une part, un recouvrement presque total entre les scans, ce qui permet d'éviter que des descripteurs valides présents dans le scan $\mathcal{P}$ ne soient plus observés dans le scan $\mathcal{Q}$ ou inversement. D'autre part, cela rend possible le calage des scans en utilisant des capteurs proprioceptifs. En effet, dans le cas de l'odométrie par exemple, même si les mesures sont sujettes à une dérive et qu'elles sont faussées par le dérapage des roues, on peut supposer que, pour de très courtes distances, elles peuvent fournir un estimé précis du déplacement du robot. Une autre méthode de calage pourrait aussi être utilisée, comme par exemple l'algorithme ICP avec un grand nombre d'itérations et de redémarrages aléatoires. L'odométrie peut toutefois, si disponible, être utilisée comme raccourci pour cette étape. Le reste de la méthode n'en dépend aucunement. Elle peut donc être arbitrairement imprécise par la suite.

Les correspondances entre les scans $\mathcal{P}$ et $\mathcal{Q}$ sont établies selon un critère du plus proche voisin dans l'espace des caractéristiques des descripteurs. Elles sont ensuite validées, en fonction du calage des scans, selon la proximité des points correspondants dans l'espace physique. Pour définir plus précisément la méthode, soient $\mathbf{f}_{\mathcal{P},i}$ et $\mathbf{f}_{\mathcal{Q},k}$ deux descripteurs ainsi appariés. Les coordonnées des points correspondants à ces descripteurs, avec l'origine définie à la position du scanner laser dans $\mathcal{P}$, sont $\mathbf{p}_i$ et $\mathbf{q}_k$. Le descripteur $\mathbf{f}_{\mathcal{P},i}$ du nuage de points $\mathcal{P}$ est étiqueté *positif connu* si les points $\mathbf{p}_i$ et $\mathbf{q}_k$ sont suffisamment proches dans l'espace physique, selon une métrique [8] qui tient compte du modèle de bruit du scanner laser. Les deux critères utilisés sont l'erreur angulaire ψ_{err} entre les points :

$$\psi_{err} = \cos^{-1}\left(\frac{\mathbf{p}_i \cdot \mathbf{q}_k}{\|\mathbf{p}_i\|\|\mathbf{q}_k\|}\right) < \psi_{max} \tag{4.3}$$

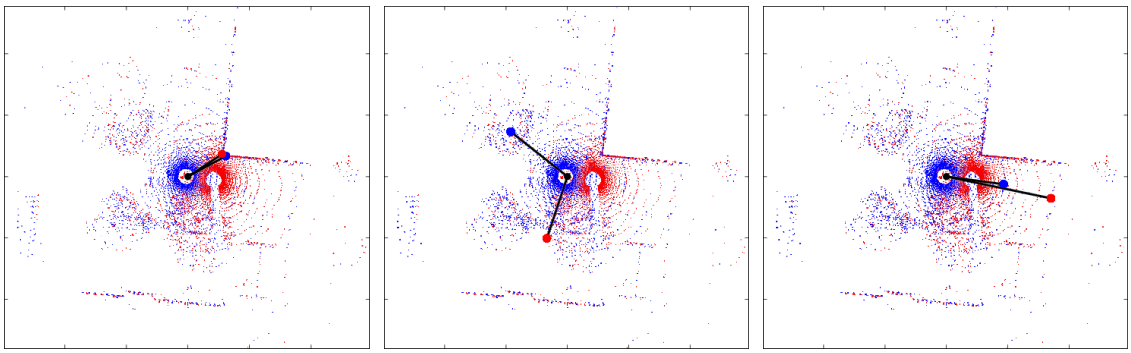
et l'erreur en distance d_{err} par rapport à l'origine :

$$d_{err} = |\|\mathbf{p}_i\| - \|\mathbf{q}_k\|| < d_{max}. \tag{4.4}$$

Les seuils ψ_{max} et d_{max} sont définis en fonction du modèle de bruit du scanner laser et en tenant compte de la faible erreur de calage. Il faut préciser que les descripteurs du scan $\mathcal{Q}$ ne sont pas inclus dans l'ensemble de données d'entraînement. Ceux-ci servent uniquement à identifier les descripteurs fiables dans $\mathcal{P}$. La Figure 4.11 illustre par quelques exemples cette méthode de validation des correspondances.

Pour ce qui est des descripteurs ne respectant pas l'un ou l'autre des critères ci-haut, nous soutenons qu'il serait inexact de les considérer comme des exemples négatifs pour l'entraînement. En effet, plusieurs facteurs peuvent mener un descripteur stable et distinctif à établir

8. Cette métrique est en fait une approximation du modèle de bruit du capteur, valide pour des positions rapprochées du scanner laser.



(a) Correspondance valide. (b) Correspondance erronée. (c) Correspondance erronée.

FIGURE 4.11: Illustration de la validation des correspondances effectuée pour étiqueter les descripteurs, en utilisant les scans 22 et 27 de *Hannover*. Les gros points rouge et bleu sont des points correspondants. Le point noir au centre donne la position de l'origine.

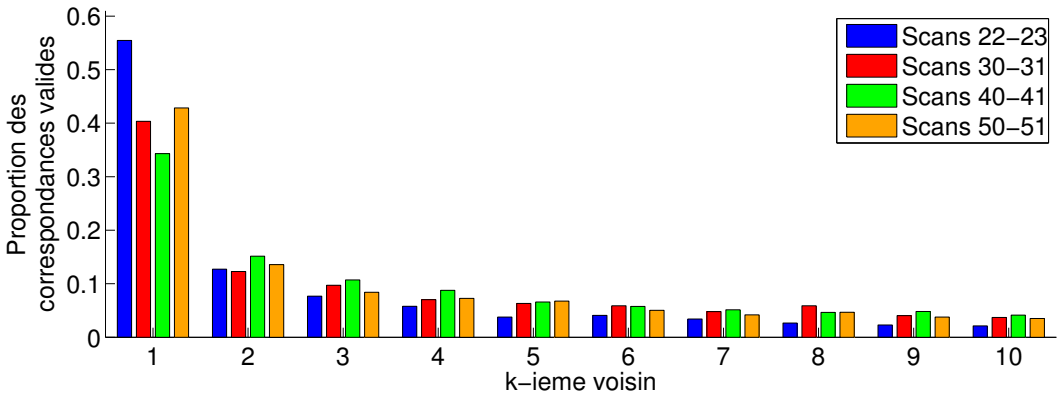


FIGURE 4.12: Proportion des correspondances valides qui sont établies rendu au k-ième voisin, pour les dix plus proches voisins, entre quelques paires de scans consécutifs de *Hannover* en utilisant les descripteurs FPFH ($r = 1$ m). La proportion des correspondances valides diminue graduellement en fonction de k.

une correspondance erronée. Par exemple, le bruit inévitable dans les points mesurés peut faire en sorte que des descripteurs situés au même endroit dans l'environnement aient des valeurs caractéristiques légèrement différentes d'un scan à l'autre. Il est donc possible que le plus proche voisin d'un descripteur (dans l'espace des caractéristiques) n'établisse pas une correspondance valide, mais que le deuxième ou le troisième le fasse. Cette affirmation est vérifiée par le graphique de la Figure 4.12, montrant la proportion des correspondances valides qui sont établies rendu au k-ième voisin, pour les dix plus proches voisins, entre quelques paires de scans consécutifs de *Hannover* en utilisant les FPFH. Cette situation est bien prise en compte par l'algorithme SAC-IA, étant donné que les correspondances sont pigées au hasard parmi les k-PPV des descripteurs. Un deuxième facteur est que, même si les points de vue des scans d'entraînement sont rapprochés, il est possible que certains objets

de l'environnement ne soient pas observés dans les deux scans. Ceci peut être causé, entre autres, par une région tombant en dehors de la portée du scanner laser ou par l'occlusion d'une région par un autre objet lors du petit déplacement du robot. Dans ces cas, même si les descripteurs situés dans ces régions sont stables et distinctifs, les correspondances ne peuvent pas être établies. Pour ces raisons, les descripteurs échouant le test de validation des équations (4.3) et (4.4) sont considérés comme des exemples *non étiquetés*, étant donné qu'ils peuvent contenir des exemples *positifs cachés*.

L'ensemble de données d'entraînement contient donc les descripteurs du scan $\mathcal{P}$, qui sont soit étiquetés positifs connus, soit non étiquetés. L'apprentissage *positive and unlabeled*, présenté à la section suivante, est un paradigme d'apprentissage automatique semi-supervisé [9] qui est bien adapté à ce type de problème.

4.4.2 Apprentissage *positive and unlabeled*

L'apprentissage *positive and unlabeled* (*PU learning*) [13] est une variante de l'apprentissage semi-supervisé qui est bien adaptée aux situations où les données d'entraînement sont disponibles sous la forme d'exemples positifs et d'exemples non étiquetés. Celle-ci tient compte du fait que l'ensemble des exemples non étiquetés contient à la fois des exemples positifs cachés et des exemples négatifs pour entraîner un classificateur des exemples positifs et négatifs.

Soit $S \subseteq X \times L = \{(\mathbf{x}_1, s_1), (\mathbf{x}_2, s_2), ..., (\mathbf{x}_m, s_m)\}$ un ensemble de m données d'entraînement, où X est l'espace d'entrée (caractéristiques) des exemples et $L \in \{l, u\}$. Les exemples pour lesquels $s = l$ sont des exemples étiquetés et ceux pour lesquels $s = u$ sont des exemples non étiquetés. Soit de plus une étiquette $y_i \in \{0, 1\}$, possiblement inconnue, à laquelle chaque exemple est également associé. Un exemple est positif si $y = 1$ et négatif si $y = 0$. Dans le contexte de l'apprentissage *positive and unlabeled*, uniquement les exemples positifs sont étiquetés, c'est-à-dire que si $s = l$, alors $y = 1$. Pour les exemples dont $s = u$, l'étiquette est $y \in \{0, 1\}$. L'objectif est d'entraîner un prédicteur $h(\mathbf{x}) = p(y = 1 \mid \mathbf{x})$ à partir de l'ensemble d'exemples S, à l'aide d'un algorithme d'apprentissage supervisé. Pour ce faire, il faut donc prendre en considération que des exemples positifs peuvent se cacher parmi l'ensemble des exemples non étiquetés.

Des algorithmes tels que One-Class SVM [39] s'adressent à ce problème d'apprentissage en utilisant uniquement les exemples positifs connus pour entraîner un estimateur $p(y = 1 \mid \mathbf{x})$. Cependant, contrairement au *PU learning*, cette approche n'utilise pas toute l'information disponible dans les données d'entraînement, car l'information contenue dans les données non étiquetées est ignorée par l'algorithme. Ce qui suit présente une méthode proposée par Elkan

9. Un apprentissage est dit semi-supervisé lorsque l'ensemble d'entraînement contient à la fois des exemples étiquetés et des exemples non étiquetés.

et Noto [13], qui permet d'adapter au *PU learning* n'importe quel algorithme d'apprentissage supervisé produisant un classificateur probabiliste (modèle génératif, voir section 3.1.1).

Renforçons le fait qu'un exemple négatif a une probabilité nulle de se retrouver dans l'ensemble des exemples étiquetés. Formellement, cet énoncé peut se traduire par :

$$p(s = l \mid \mathbf{x}, y = 0) = 0 \tag{4.5}$$

Dans le cas des exemples positifs, ceux-ci se retrouvent dans l'ensemble des exemples étiquetés avec une certaine probabilité. L'hypothèse est faite que cette probabilité est une constante c identique pour tous les exemples, c'est-à-dire :

$$p(s = l \mid \mathbf{x}, y = 1) = p(s = l \mid y = 1) = c \tag{4.6}$$

Sous de telles conditions, il est possible de démontrer qu'un prédicteur probabiliste $g(\mathbf{x})$, entraîné avec un algorithme standard d'apprentissage supervisé à partir de l'ensemble S tel que $g(\mathbf{x}) \approx p(s = l \mid \mathbf{x})$, prédit des probabilités qui diffèrent d'un facteur c des probabilités recherchées $p(y = 1 \mid \mathbf{x})$. En effet,

$$\begin{aligned} p(s = l \mid \mathbf{x}) &= p(y = 1 \wedge s = l \mid \mathbf{x}) && (4.7)\\ &= p(y = 1 \mid \mathbf{x}) \cdot p(s = l \mid y = 1, \mathbf{x}) && (4.8)\\ &= p(y = 1 \mid \mathbf{x}) \cdot p(s = l \mid y = 1) && (4.9)\\ &= p(y = 1 \mid \mathbf{x}) \cdot c && (4.10) \end{aligned}$$

Donc :

$$h(\mathbf{x}) = p(y = 1 \mid \mathbf{x}) = \frac{p(s = l \mid \mathbf{x})}{c} \approx \frac{g(\mathbf{x})}{c} \tag{4.11}$$

Les auteurs soulèvent deux conséquences de ce résultat. La première est que la probabilité retournée par le prédicteur $h(\mathbf{x})$ est nécessairement supérieure à celle de $g(\mathbf{x})$. L'adaptation au *PU learning* ajuste (à la hausse) les probabilités des données à être positives, mais ne change pas l'ordre de probabilités des données. Autrement dit, une donnée plus probable d'être positive qu'une autre au départ reste nécessairement plus probable après l'ajustement par la constante c. La deuxième conséquence est que $h(\mathbf{x})$ est une probabilité bien définie $h(\mathbf{x}) \leq 1$ seulement si $g(\mathbf{x}) \leq c$. Cela implique qu'il est impossible que $g(\mathbf{x}) > c$.

La valeur de la constante c peut être estimée à partir d'un ensemble de données $V \subseteq S$ exclu au départ des données d'entraînement. Un prédicteur probabiliste $g(\mathbf{x})$ est entraîné avec $S \backslash V$, puis utilisé pour calculer un estimateur de la probabilité qu'une donnée positive soit étiquetée :

$$\hat{c}_1 = \frac{1}{|P|} \sum_{\mathbf{x} \in P} g(\mathbf{x}) \tag{4.12}$$

où P est l'ensemble de tous les exemples positifs connus dans V. Il faut noter que, puisque les exemples dans V n'ont pas été utilisés pour entraîner $g(\mathbf{x})$, l'estimateur est non biaisé. Un

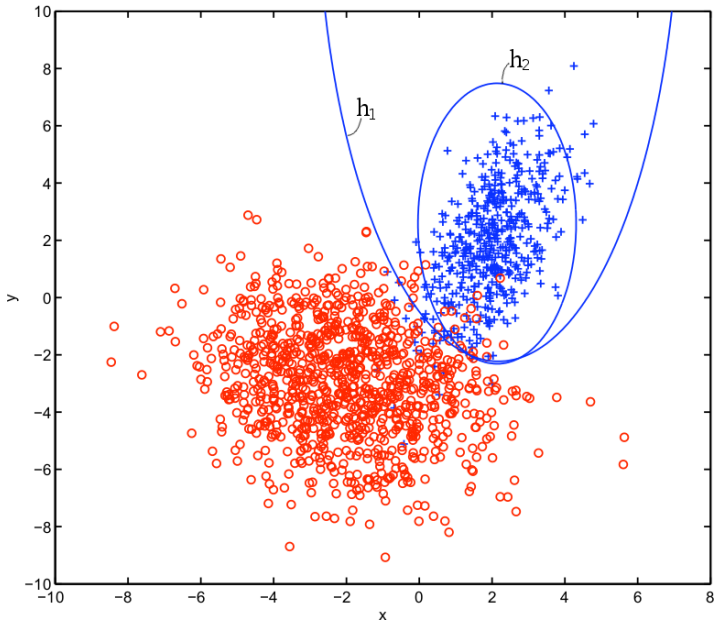


FIGURE 4.13: Illustration de la méthode d'apprentissage *positive and unlabeled*. La grande et la petite ellipse montrent la frontière de précision de classificateurs entraînés en utilisant, respectivement, tous les exemples étiquetés (h_1) et 20% des exemples positifs comme étiquetés (h_2). Image tirée de [13].

deuxième estimateur, similaire au premier, peut être obtenu par l'équation :

$$\hat{c}_2 = \frac{\sum_{\mathbf{x} \in P} g(\mathbf{x})}{\sum_{\mathbf{x} \in V} g(\mathbf{x})} \tag{4.13}$$

Un troisième, basé sur le fait que $g(\mathbf{x}) \leq c$ pour tout $\mathbf{x}$, est :

$$\hat{c}_3 = \max_{\mathbf{x} \in V} g(\mathbf{x}) \tag{4.14}$$

Les auteurs mentionnent que le premier estimateur à l'équation (4.12) est préférable en pratique car sa variance est la plus faible.

La Figure 4.13, tirée de [13], illustre les performances du *PU learning* pour un ensemble de données contenant 500 données positives et 1000 négatives. Deux classificateurs ont été entraînés : le premier, $h_1(\mathbf{x})$, en utilisant toutes les données et le deuxième, $h_2(\mathbf{x})$, en utilisant 20% des données positives comme exemples positifs connus (étiquetés) et toutes les autres comme exemples non étiquetés. La grande ellipse montre la frontière de décision de $h_1(\mathbf{x})$ et la plus petite celle de $h_2(\mathbf{x})$. La constante c a été estimée, par l'équation (4.12), à la valeur 0.1928 en utilisant seulement 20 données étiquetées, ce qui est très près de la valeur réelle 0.2. Le classificateur obtenu par *PU learning* sur les données partiellement étiquetées a sensiblement les mêmes performances que celui entraîné sur toutes les données étiquetées. Cette méthode permet donc d'obtenir un classificateur d'exemples positifs et

négatifs, entraîné seulement avec des exemples positifs et non étiquetés en utilisant un algorithme d'apprentissage supervisé (génératif) standard.

4.4.3 Algorithme d'apprentissage proposé

Dans l'ensemble d'entraînement généré par le robot, les exemples se présentent sous la forme d'exemples positifs connus et d'exemples non étiquetés. Connaissant l'existence de la méthode d'apprentissage *positive and unlabeled* présentée ci-haut, il apparaît donc naturel d'aborder le problème d'apprentissage sous ce paradigme.

Il est toutefois pertinent de justifier que l'hypothèse de l'équation 4.6, à la base de cette méthode, est respectée. Par définition, les descripteurs stables et distinctifs sont fiables et établissent des correspondances valides. Ceux-ci devraient donc normalement être étiquetés positifs connus ($s = l$) dans l'ensemble d'entraînement. Pour deux scans acquis à des positions rapprochées, comme c'est le cas des scans utilisés pour la génération des données d'entraînement, la principale raison pour laquelle des descripteurs fiables peuvent mener à des correspondances erronées est le bruit de mesure. Or, ce bruit de mesure est aléatoire, ce qui permet de supposer qu'un sous-ensemble aléatoire des descripteurs fiables mène à des correspondances erronées. Par conséquent, on peut supposer que la probabilité pour un descripteur fiable d'être étiqueté positif connu est indépendante du descripteur lui-même.

L'algorithme recherché est un algorithme d'apprentissage supervisé pouvant être adapté au *PU learning*. Cet algorithme doit donc produire un prédicteur probabiliste $g(\mathbf{x}) = p(s = 1 \mid \mathbf{x})$. De plus, les nuages de points pouvant contenir des centaines de milliers de points, l'algorithme se doit d'être rapide et applicable aux très grands ensembles de données. Cette propriété est importante pour le problème de classification de descripteurs, car les applications réelles en robotique mobile visent généralement une opération en temps réel. Enfin, l'algorithme doit être robuste au bruit de mesure des points, qui se répercute comme bruit dans les valeurs caractéristiques des descripteurs.

Supposons que la proportion d'exemples positifs cachés dans l'ensemble des exemples non étiquetés U est γ. Lorsque la valeur de γ augmente, le bruit introduit par les exemples positifs cachés dans U augmente également. Par conséquent, on peut s'attendre à ce que les prédicteurs entraînés par un algorithme supervisé soient affectés par la valeur de γ. En se basant sur cette observation, Mordelet et Vert [26][27] proposent d'utiliser le bagging [5] (section 3.2.5) pour apprendre des prédicteurs à partir d'exemples positifs et non étiquetés. À titre de rappel, le bagging consiste à combiner les prédictions, par un vote de majorité ou autre, de plusieurs prédicteurs entraînés en utilisant des ensembles de données d'entraînement différents obtenus par bootstrapping. Tel qu'indiqué par Breiman [6], cette méthode fonctionne bien lorsque les prédicteurs ne sont pas corrélés. L'échantillonnage aléatoire de U avec remise est susceptible de mener à des valeurs différentes de γ. Par conséquent, tel que mentionné par Mordelet

et Vert, on peut s'attendre à obtenir des prédicteurs non corrélés, pour lesquels le bagging devrait améliorer les performances. Ces auteurs proposent un bagging de SVM [27] pour apprendre un classificateur à partir d'exemples positifs et d'exemples non étiquetés. Cette méthode est cependant gourmande en temps de calcul, étant donnée la complexité algorithmique élevée de SVM en fonction du nombre de données, et n'a donc pas été considérée pour le problème d'apprentissage de descripteurs.

Nous reprenons toutefois l'idée de Mordelet et Vert de faire du bagging, ce qui peut s'avérer utile également pour augmenter la robustesse au bruit dans les valeurs caractéristiques des descripteurs. Par conséquent, l'algorithme que nous proposons d'adapter au *PU learning* est une version légèrement modifiée de l'algorithme random forest (section 3.2.6), basé en partie sur le bagging. Ce dernier a l'avantage d'être rapide et de pouvoir s'étendre à des ensembles de données très volumineux. Même si, typiquement, random forest ne produit pas un classificateur probabiliste, il est possible d'estimer $p(s = l \mid x)$. Dans l'implémentation de notre approche, ceci est fait en calculant la moyenne des probabilités de classification sur tous les arbres de décision. La probabilité retournée pour un arbre de décision correspond à la proportion d'exemples étiquetés positifs connus ($s = l$) parmi les exemples appartenant à la feuille de l'arbre dans laquelle tombe le nouvel exemple à classer.

La proportion de descripteurs fiables dans les environnements naturels s'avère en général minoritaire [10]. Ce débalancement entre les classes d'exemples positifs connus et d'exemples non étiquetés peut causer problème lors de l'entraînement. Afin d'être adapté à cette particularité du problème d'apprentissage des descripteurs dans ce type d'environnement, l'algorithme de random forest proposé diffère sensiblement de celui présenté originellement dans sa phase de bootstrapping. Plutôt que d'échantillonner aléatoirement (avec remise) $|P \cup U|$ exemples dans $P \cup U$, l'algorithme échantillonne aléatoirement (avec remise) $|P|$ exemples dans P et $|P|$ exemples dans U. Cette approche est similaire à celle proposée par Mordelet et Vert [26][27], dans le sens où les classes fournies à l'algorithme d'apprentissage sont balancées et que cela réduit le temps d'entraînement lorsque $|P| \ll |U|$, ce qui est le cas ici.

4.4.4 Filtrage des scans

Une fois le classificateur entraîné en utilisant l'algorithme d'apprentissage proposé, celui-ci est utilisé pour prédire les étiquettes des descripteurs des nouveaux scans. Ceci est effectué immédiatement après le calcul des descripteurs. Les descripteurs classés positifs, de même que les points auxquels ils ont été calculés, sont conservés pour les étapes d'estimation de correspondances et de calage. Les autres descripteurs et points sont éliminés. Le nombre de points dans les scans filtrés est donc le même que le nombre de descripteurs positifs

10. Selon nos expérimentations, cette proportion est en général inférieure à 20% pour l'ensemble de données *Hannover*. Pour l'ensemble de données *Wood Summer* (présenté au chapitre 5), plus dense en végétation, elle est inférieure à 10%.

conservés.

Cette réduction du nombre de points dans les scans filtrés permet d'accélérer le processus de calage fait par SAC-IA. En effet, le temps de calcul de la qualité du calage, effectué à chaque itération, dépend du nombre de points dans les scans. Il serait aussi possible de conserver tous les points et de n'utiliser que les descripteurs classés positifs pour établir des correspondances. L'évaluation de la qualité du calage effectué à chaque itération serait alors basée sur tous les points des scans. Celle-ci serait donc plus précise, mais le gain en temps de calcul serait nul (pour un même nombre d'itérations).

Chapitre 5

Expérimentations et résultats

Ce chapitre présente et analyse les résultats des expérimentations effectuées pour tester la méthode de filtrage proposée. La structure des tests réalisés est d'abord détaillée à la section 5.1, suivie des résultats d'entraînement, de filtrage et de calage obtenus à la section 5.2.

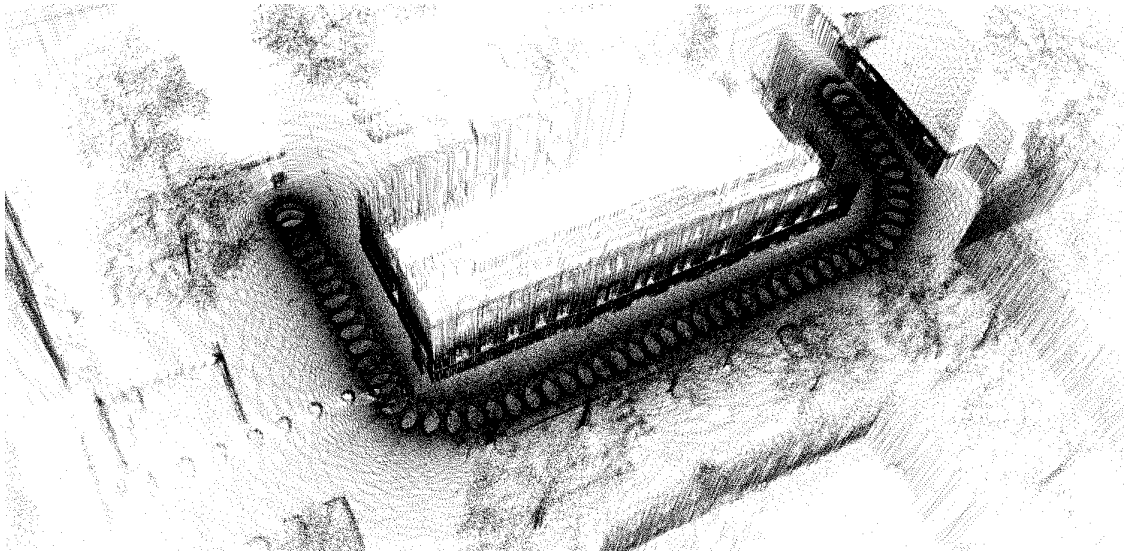
Toutes les expérimentations ont été réalisées en utilisant les descripteurs FPFH. Leur dimensionnalité relativement faible [1] (33 dimensions) comparée à celle d'autres types de descripteurs comme les SHOT (352 dimensions) en fait des candidats parfaits pour notre approche. De plus, le calcul des FPFH dépend principalement d'un seul paramètre, qui est l'intuitif rayon r délimitant le voisinage. Ces descripteurs sont donc également moins sensibles à l'ajustement de paramètres. Les SHOT n'ont pas été utilisés, même s'ils présentent des performances similaires aux FPFH, car le temps requis pour les calculer et les comparer est significativement plus long que celui des FPFH.

5.1 Structure de tests

L'approche proposée a été testée pour deux séquences de nuages de points 3D extraits d'ensembles de données disponibles publiquement. La première séquence est constituée des scans 22 à 80 de *Hannover*, comprenant plusieurs arbres et buissons de même que des bâtiments. Ces scans couvrent une distance d'environ 84 m et contiennent 13 557 points en moyenne. La raison pour laquelle la séquence débute au scan 22 est que le positionnement réel du robot à chaque scan, utilisé pour évaluer la qualité des résultats de calage, n'était pas disponible pour les scans 1 à 21. Cette séquence de scans est présentée à la Figure 5.1(a). La deuxième séquence de test est constituée des scans de l'ensemble de données *Wood Summer* [31]. Cet ensemble de données comprend 36 scans acquis dans un environnement non structuré et dense en végétation. Le scan 0 de cette séquence est présenté à la Figure 5.1(b).

1. Noter que ces valeurs de dimensionnalité sont celles des FPFH et des SHOT tels qu'implémentés par défaut dans la librairie *Point Cloud Library* (PCL) [36], utilisée pour les expérimentations. La dimensionnalité peut changer selon certains paramètres.

Ces scans suivent une trajectoire d'environ 21 m et contiennent en moyenne 182 000 points.



(a) *Hannover* - scans 22 à 80.

(b) *Wood Summer* - uniquement le scan 0 est montré.

FIGURE 5.1: Visualisation des séquences de test utilisées. La séquence des scans 22 à 80 de *Hannover* (5.1(a)) présente plusieurs arbres, buissons et bâtiments. La séquence *Wood Summer*, dont le scan 0 est montré à la figure 5.1(b), a été acquise dans un environnement dense en végétation.

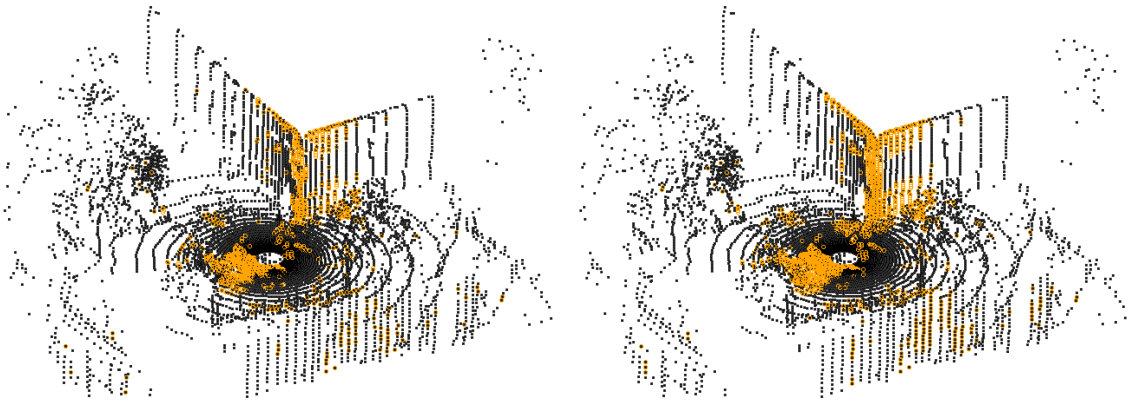
Pour chacune de ces séquences de test, un classificateur a été entraîné une fois seulement, selon la méthode décrite à la section 4.4, en utilisant les deux premiers scans de la séquence. Le classificateur entraîné a ensuite été utilisé pour filtrer les descripteurs des scans pour tout le reste de la séquence, sans entraînement supplémentaire. Ceci permet de vérifier qu'un

classificateur entraîné à un certain endroit dans l'environnement continue de donner des résultats satisfaisants à des emplacements différents, à mesure que le robot se déplace, tant que l'environnement reste similaire. Il est à noter que, dans le cadre de cet ouvrage, ce critère de similarité n'a pas été défini mathématiquement. Il s'agit plutôt d'un critère qualitatif basé sur la présence d'objets similaires dans l'environnement, comme par exemple une même espèce d'arbre ou d'arbuste, ou encore une même densité de végétation. Il serait toutefois intéressant d'être en mesure de quantifier cette similarité, ce qui permettrait de détecter des changements de types d'environnement et de déclencher un nouvel entraînement. Cette option est abordée plus loin dans la discussion sur les travaux futurs.

Il est ardu de quantifier les erreurs de classification commises par l'algorithme d'apprentissage proposé, car les vraies étiquettes $y \in \{0, 1\}$ des descripteurs non étiquetés dans l'ensemble d'entraînement (pour lesquels $s = u$) sont inconnues. Par conséquent, une façon indirecte d'évaluer la qualité du filtrage réalisé est d'effectuer des tests de calage. L'amélioration du calage est d'ailleurs l'objectif de la méthode proposée. L'algorithme SAC-IA (section 2.3.2) a donc été utilisé pour effectuer, pour chaque paire de scans consécutifs, plusieurs essais de calage. Le fait de tester plusieurs fois le calage pour les mêmes scans permet de prendre en compte la nature stochastique de cet algorithme.

Afin de bien cerner l'effet de la méthode de filtrage proposée, les performances des méthodes suivantes sont comparées pour le calage :
– En utilisant tous les descripteurs ;
– En utilisant les descripteurs fiables uniquement (classés positifs) ;
– En utilisant un sous-ensemble aléatoire de N_{rand} descripteurs.
La troisième approche, une façon valide d'accélérer le calage des nuages de points, a été incluse pour démontrer que la méthode de filtrage proposée augmente bel et bien la proportion de descripteurs fiables dans l'ensemble de descripteurs filtrés. La valeur de N_{rand} a été fixée, pour chacune des séquences de test, au nombre moyen de descripteurs classés positifs sur toute la séquence. Ce choix assure une comparaison équitable avec la méthode de filtrage proposée, puisqu'il fait en sorte que leurs temps de calcul sont similaires. Il faut préciser que chaque essai individuel de calage utilise un sous-ensemble différent de N_{rand} descripteurs échantillonnés aléatoirement.

Les trois différentes approches ont été évaluées et comparées en utilisant la précision des calages réalisés, en termes d'erreurs en rotation et en translation. Comme la direction des erreurs n'a pas d'importance, la norme de ces erreurs a été utilisée. Les temps de calculs ont également été comparés, afin d'estimer l'accélération de la méthode proposée.



(a) Données d'entraînement. (b) Test sur données d'entraînement.

FIGURE 5.2: Résultat de la génération des données d'entraînement en utilisant les scans 22 et 23 de *Hannover* (à gauche) et résultat de la classification des descripteurs utilisés pour l'entraînement (à droite). Les descripteurs utilisés sont les FPFH ($r = 1\,\text{m}$). Les points encerclés en orange localisent les descripteurs positifs connus (à gauche) et classés positifs (à droite). On peut voir que le classificateur réussit à bien retrouver les descripteurs qui ont été étiquetés positifs connus.

5.2 Résultats

Les sous-sections qui suivent présentent, dans l'ordre, les résultats d'entraînement, de filtrage et de calage obtenus.

5.2.1 Entraînement du classificateur

Pour la séquence de test de *Hannover*, les données d'entraînement ont été générées à partir des scans 22 et 23. Les valeurs de d_{err} et ψ_{err} utilisées pour valider les correspondances (voir les équations (4.3) et (4.4)) sont 0.5 m et 5° respectivement. Ces valeurs ont été choisies telles qu'elles sont supérieures au bruit de mesure du scanner laser. Ceci permet d'être plus permissif sur la validation des correspondances et, par conséquent, d'obtenir plus de descripteurs classés positifs par la suite. Le rayon de voisinage pour le calcul des descripteurs a été fixé à $r = 1\,\text{m}$. Cette valeur s'est avérée optimale lors de tests de correspondances réalisés (sans apprentissage) avec les FPFH sur cette séquence de test. La Figure 5.2(a) présente le résultat de la génération des données d'entraînement avec ces paramètres. La localisation des descripteurs étiquetés positifs connus (dans le scan 22) est donnée par les cercles orange. L'ensemble d'exemples d'entraînement résultant contient 1981 exemples positifs connus et 10374 exemples non étiquetés (16% de positifs connus).

L'algorithme de random forest a été entraîné en utilisant 100 arbres de décision. Selon des tests préliminaires, il ne semble pas y avoir de différence significative dans les performances de l'apprentissage au-delà de ce nombre. Le classificateur obtenu a été adapté au *PU learning*

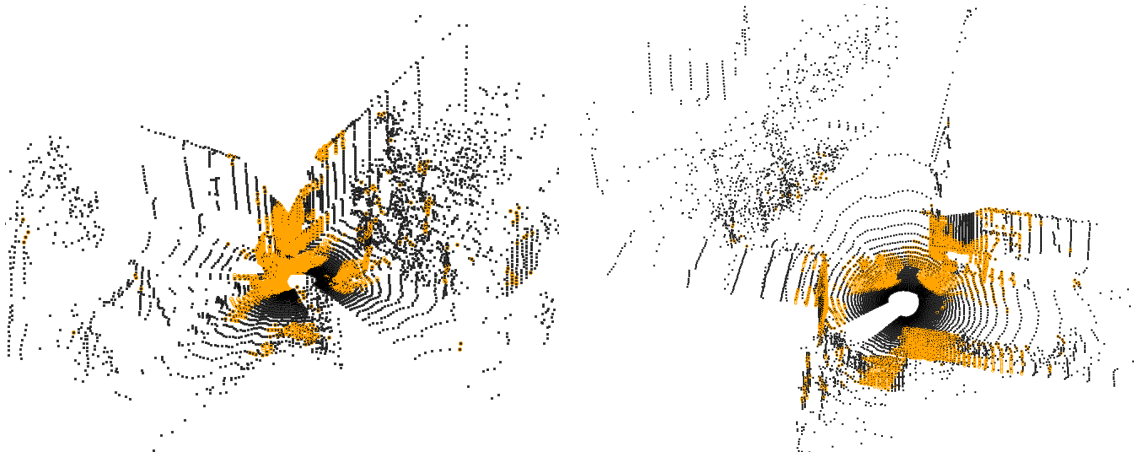
en utilisant l'estimateur $\hat{c}_1$, donné à l'équation (4.12). La valeur de $c = p(s = l \mid y = 1)$ est dans ce cas estimée à 0.77 (en utilisant 200 exemples). Pour des fins de comparaison visuelle, la Figure 5.2(b) présente le résultat de la classification des descripteurs sur le scan 22 (test sur les données d'entraînement). Le nombre de descripteurs classés positifs est de 2735, alors que le nombre de descripteurs classés négatifs est de 9620. Parmi les descripteurs étiquetés positifs connus au départ, 97.9% ont été classés positifs. Le nombre de descripteurs conservés est réduit significativement à 22.1% $(2735/(2735 + 9620))$. Il est impossible de calculer le taux de succès exact de classification du prédicteur, car les étiquettes $y \in \{0, 1\}$ des descripteurs non étiquetés sont inconnues.

Dans le cas de la séquence *Wood Summer*, l'entraînement a été effectué avec les scans 0 et 1 en utilisant les paramètres $d_{err} = 0.2\,\text{m}$ et $\psi_{err} = 2^\circ$. Le rayon de voisinage pour le calcul des descripteurs est $r = 1\,\text{m}$. Ces valeurs ont été déterminées de la même manière que pour la séquence précédente. L'ensemble d'entraînement généré comprend 10 175 exemples positifs connus et 173 004 exemples non étiquetés (5.6% de positifs connus). L'algorithme de random forest a été entraîné encore une fois en utilisant 100 arbres de décision. La valeur de c est dans ce cas estimée à 0.87 (en utilisant 1000 exemples). En testant le classificateur sur le scan 0, 20 285 descripteurs ont été classés positifs et 162 894 ont été classés négatifs. Les images de résultats ne sont pas présentées pour cette séquence de test, car le nombre élevé de points et la présence d'une grande quantité de feuillage dans les scans rend la visualisation difficile. La proportion des descripteurs étiquetés positifs connus qui ont été classés positifs par le prédicteur est de 99.8% et le nombre de descripteurs conservés est réduit à 11.1% $(20285/(20285 + 162894))$.

5.2.2 Analyse préliminaire du filtrage des descripteurs

La Figure 5.3 présente des exemples de scans de la séquence *Hannover*, filtrés en utilisant le classificateur entraîné. Les points correspondants aux descripteurs classés positifs sont encerclés en orange. Dans le cas du scan 40, 4741 descripteurs ont été classés positifs et 7303 ont été classés négatifs (39.4% de positifs). Dans le cas du scan 70, la classification a retourné 4393 descripteurs positifs et 9170 négatifs (32.3% de positifs). Même s'il est ardu de quantifier correctement les performances de la classification des descripteurs, on observe sur ces images que les descripteurs conservés semblent en majorité situés à des endroits distinctifs et stables de l'environnement. En particulier, beaucoup de descripteurs classés positifs sont situés sur les coins de murs, les troncs d'arbres et les humains près du centre (difficiles à voir sur ces images), de même que sur les régions voisines de ces objets (distinguable par leur proximité à ces objets).

Sur l'ensemble de la séquence *Hannover*, la proportion de descripteurs conservés après le filtrage se situe autour de 35% en moyenne. Il faut rappeler cependant que les scans de cet ensemble de données ont été acquis dans un environnement présentant une certaine structure



(a) *Hannover* - scan 40. (b) *Hannover* - scan 70.

FIGURE 5.3: Exemples de résultat de la classification des descripteurs FPFH en utilisant le prédicteur entraîné selon la méthode proposée. Les points encerclés en orange localisent les descripteurs classés positifs.

(bâtiments). Dans le cas de la séquence *Wood Summer*, la proportion de descripteurs conservés oscille entre 9% et 18%. Ce taux, nettement plus faible, témoigne de la difficulté accrue de l'utilisation des descripteurs dans cet environnement.

5.2.3 Résultats de calage

Le calage des scans a été effectué pour un nombre d'itérations de SAC-IA fixé à $N_{iter} = 100$. Le nombre de plus proches voisins d'un descripteur (dans l'espace des caractéristiques), parmi lesquels le descripteur correspondant est pigé, a été fixé à $k = 3$. Cette valeur s'est avérée optimale lors de tests de calage préliminaires utilisant tous les descripteurs des scans. Pour chaque paire de scans consécutifs, 100 essais de calage ont été effectués afin d'obtenir une distribution des résultats étant donnée la nature probabiliste de l'algorithme SAC-IA. Le positionnement réel du robot (vérité terrain) étant connu pour tous les scans dans les séquences de test, les erreurs d'alignement en translation et en rotation ont pu être calculées pour chacun des essais de calage.

Il faut rappeler que l'algorithme SAC-IA effectue un calage global des nuages de points, ce qui constitue une étape préalable à un calage local en utilisant un algorithme tel que ICP. L'objectif de SAC-IA n'est donc pas d'obtenir le calage le plus précis possible, mais de trouver des valeurs de $\mathbf{R}$ et $\mathbf{t}$ tombant dans le bassin de convergence du minimum global de la fonction d'erreur utilisée par ICP. Des erreurs de calage global plus petites impliquent donc une augmentation de la probabilité que le calage tombe dans ce bassin de convergence. Par conséquent, des calages obtenus avec SAC-IA dont les erreurs sont plus petites et de variances plus faibles indiquent une meilleure performance et robustesse.

Séquence *Hannover*

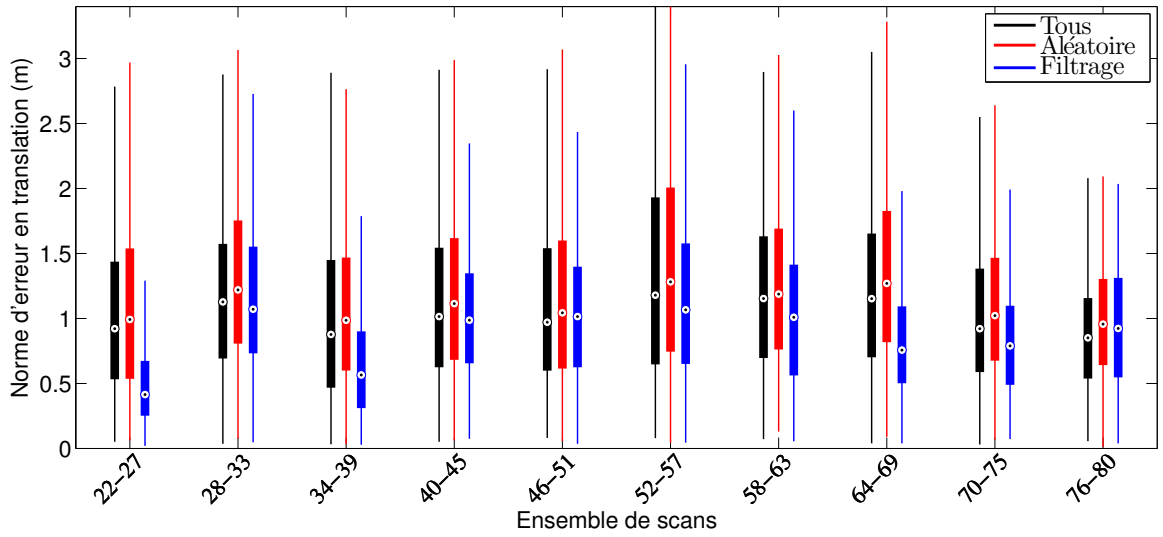
Les Figures 5.4(a) et 5.4(b) présentent, sous forme de boîtes à moustaches, les distributions des normes d'erreurs de calage, en translation et en rotation, à travers les scans de la séquence de test *Hannover*. Les résultats des calages ont été groupés par ensembles de six scans consécutifs afin d'alléger les graphiques et de mieux suivre l'évolution générale des performances de chaque méthode lorsque le robot se déplace.

Pour la méthode de sous-échantillonnage aléatoire des descripteurs, le nombre de descripteurs pigés est de $N_{rand} = 4500$. Tel qu'attendu, cette méthode de sous-échantillonnage aléatoire performe un peu moins bien que lorsque tous les descripteurs sont utilisés. Cela s'explique par le fait que l'ensemble de descripteurs conservés contient statistiquement la même proportion de descripteurs fiables qu'au départ. Le nombre réduit de descripteurs fait cependant en sorte que SAC-IA peut explorer moins de possibilités et que la qualité de l'alignement, calculée à chaque itération, est évaluée sur moins de points et est donc un peu moins précise. Pour ce qui est de la méthode de filtrage proposée, on remarque une amélioration significative de la qualité du calage, autant en rotation qu'en translation, pour les scans suivant immédiatement l'entraînement (23-27). Ces scans sont toutefois très similaires à ceux utilisés pour l'entraînement et ne peuvent donc pas être représentatifs de la capacité de généralisation du classificateur. L'amélioration par rapport à la méthode utilisant tous les descripteurs est toujours notable, même si moins significative, pour le reste de la séquence (28-80) : la médiane des erreurs (ronds blancs au centre des barres) est presque toujours plus petite, de même que la variance des erreurs qui est toujours similaire ou légèrement plus faible.

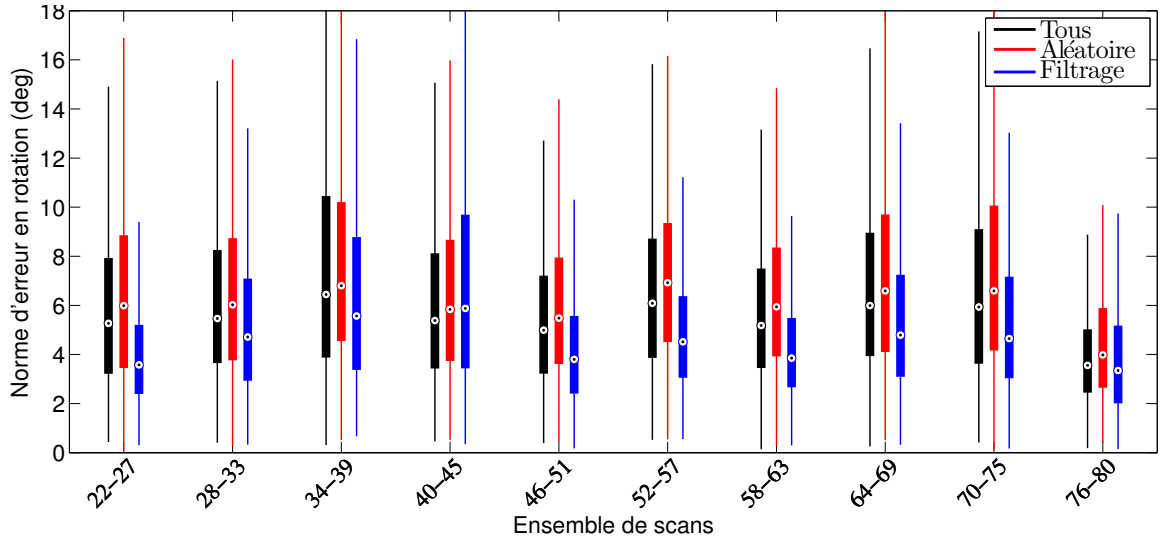
Les Figures 5.5(a) et 5.5(b) présentent les distributions, sous forme d'histogrammes, des erreurs de calage en translation et en rotation pour l'ensemble de la séquence *Hannover*. Les lignes verticales pointillées sur ces graphiques donnent la valeur médiane de chacune des distributions, alors que les lignes verticales pleines donnent la valeur du 95^e centile. Ces figures montrent plus clairement la réduction de l'erreur médiane pour l'approche proposée comparée à lorsque tous les descripteurs sont utilisés [2]. On remarque aussi que le nombre d'occurrences des erreurs importantes (plus de 2 m en translation ou plus de 10° en rotation) est réduit significativement, ce qui est indiqué par la position des 95^e centiles. Cette réduction peut être la clé de l'amélioration de la robustesse du calage global, car elle augmente la probabilité d'un algorithme d'alignement local de converger vers la vraie solution (minimum global).

Ces résultats indiquent que la méthode de filtrage proposée peut bel et bien réduire le nombre de descripteurs utilisés tout en augmentant la proportion de descripteurs fiables pour un calage grossier.

2. Voir le Tableau 5.1 présenté plus bas pour les pourcentages de réduction des erreurs.

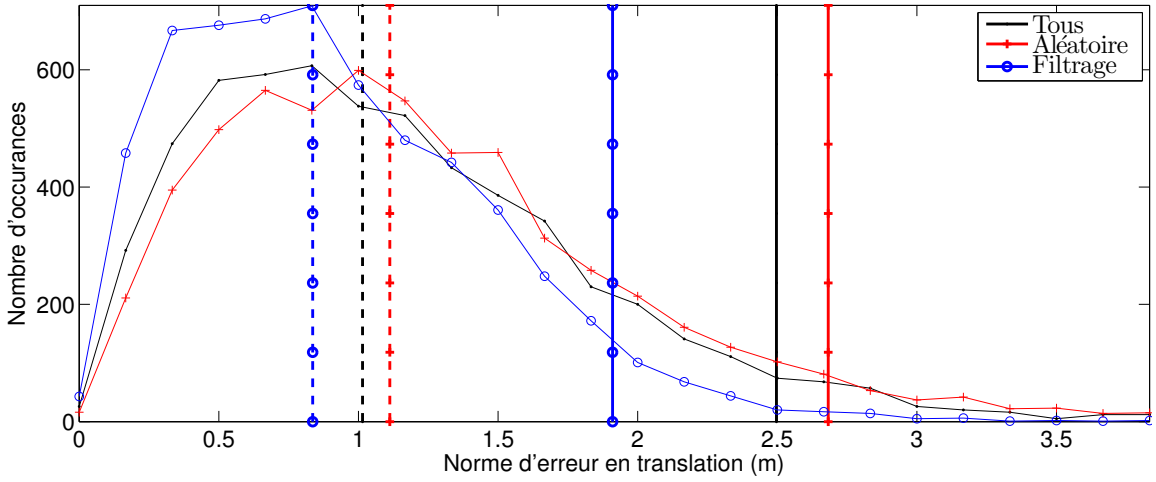


(a) Normes d'erreurs en translation pour la séquence *Hannover*.

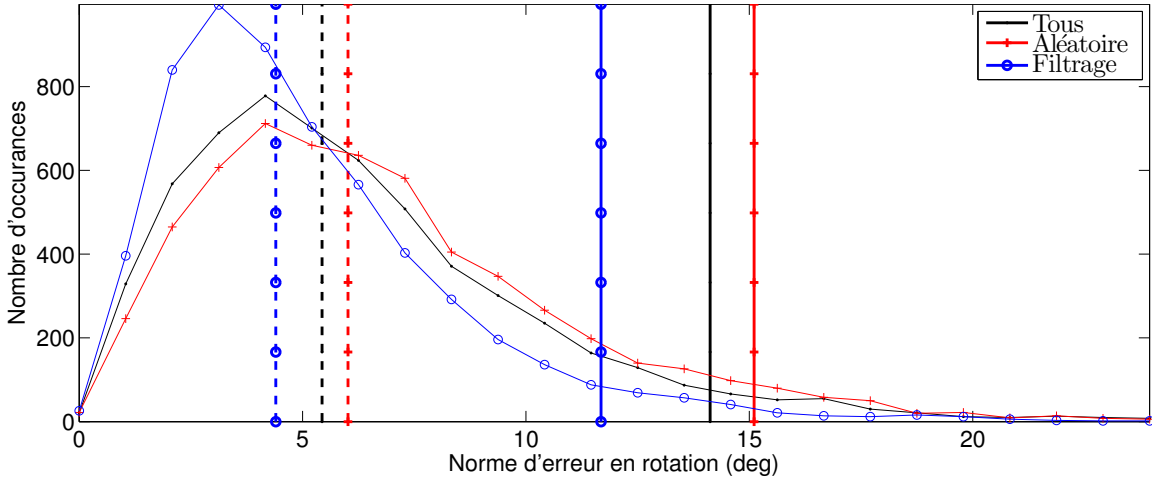


(b) Normes d'erreurs en rotation pour la séquence *Hannover*.

FIGURE 5.4: Distributions des normes d'erreurs de calage à travers les scans de la séquence *Hannover*, en translation (5.4(a)) et en rotation (5.4(b)). Les résultats ont été groupés par ensembles de six scans consécutifs afin d'alléger les graphiques et de mieux faire ressortir l'évolution générale des performances de chaque méthode à mesure que le robot se déplace.



(a) Distributions des normes d'erreurs en translation pour la séquence *Hannover*.



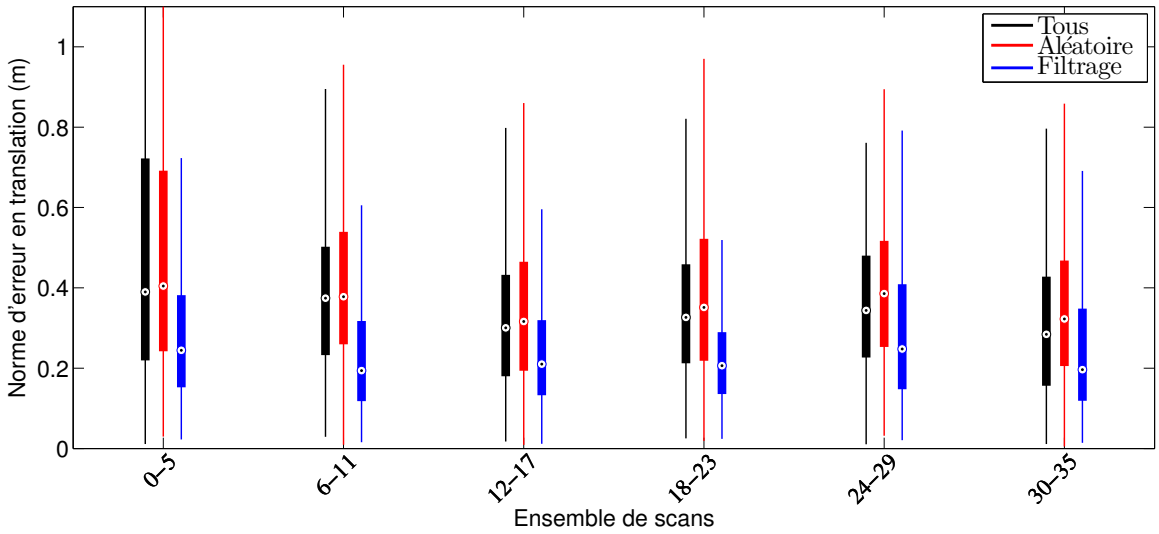
(b) Distribution des normes d'erreurs en rotation pour la séquence *Hannover*.

FIGURE 5.5: Distributions des erreurs de calage, en translation (5.5(a)) et en rotation (5.5(b)), pour l'ensemble de la séquence *Hannover*. Les lignes verticales pointillées indiquent les valeurs médianes des distributions, alors que les lignes verticales pleines indiquent celles des 95^e centiles.

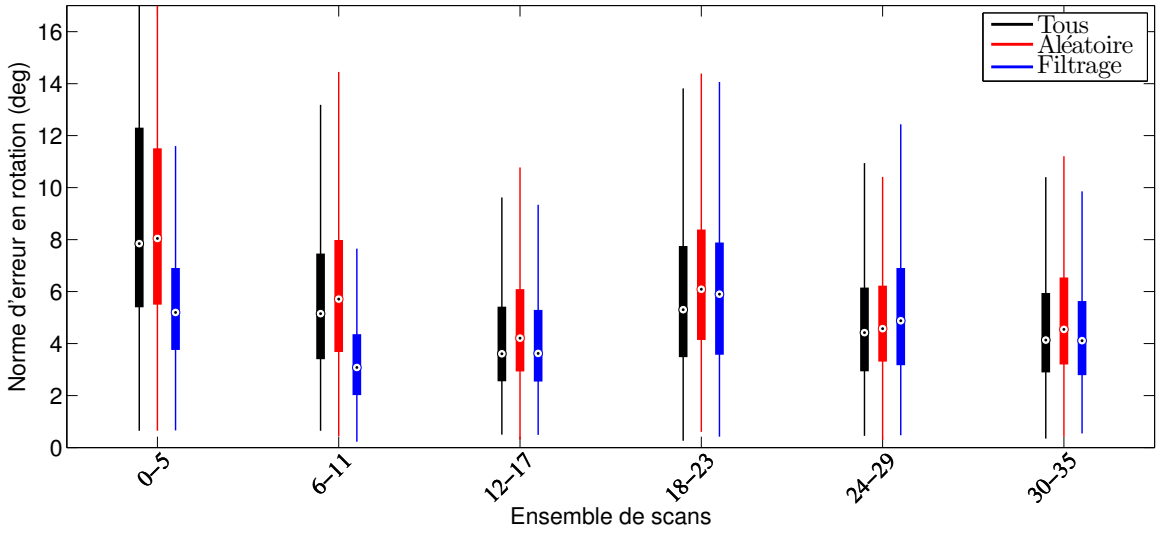
Séquence *Wood Summer*

Les Figures 5.6(a) et 5.6(b) présentent les graphiques de distributions des normes d'erreurs de calage à travers les scans de la séquence de test *Wood Summer*. Dans ce cas, le nombre de descripteurs pigés pour la méthode de sous-échantillonnage aléatoire est $N_{rand} = 25000$. Encore une fois, cette méthode de sous-échantillonnage aléatoire performe un peu moins bien que celle utilisant tous les descripteurs. En utilisant la méthode de filtrage proposée, les erreurs et variances d'erreurs en translation sont cette fois significativement plus faibles que lorsque tous les descripteurs sont utilisés. La réduction des erreurs en rotation est également importante pour le premier tiers de la séquence (scans 0-11), mais peu notable par la suite.

Les Figures 5.7(a) et 5.7(b) présentent les histogrammes de distributions d'erreurs de calage en translation et en rotation pour l'ensemble de la séquence *Wood Summer*. Pour cet environnement, l'amélioration de l'erreur en translation est particulièrement significative, avec une réduction de près de 40% de l'erreur médiane (voir le Tableau 5.1 présenté plus bas). Le nombre d'erreurs importantes (plus de 0.6 m en translation ou plus de 10° en rotation) est également significativement réduit. Cette augmentation de la robustesse et de la précision du calage indique clairement que la méthode proposée pour filtrer les descripteurs est bénéfique pour les environnements naturels non structurés.

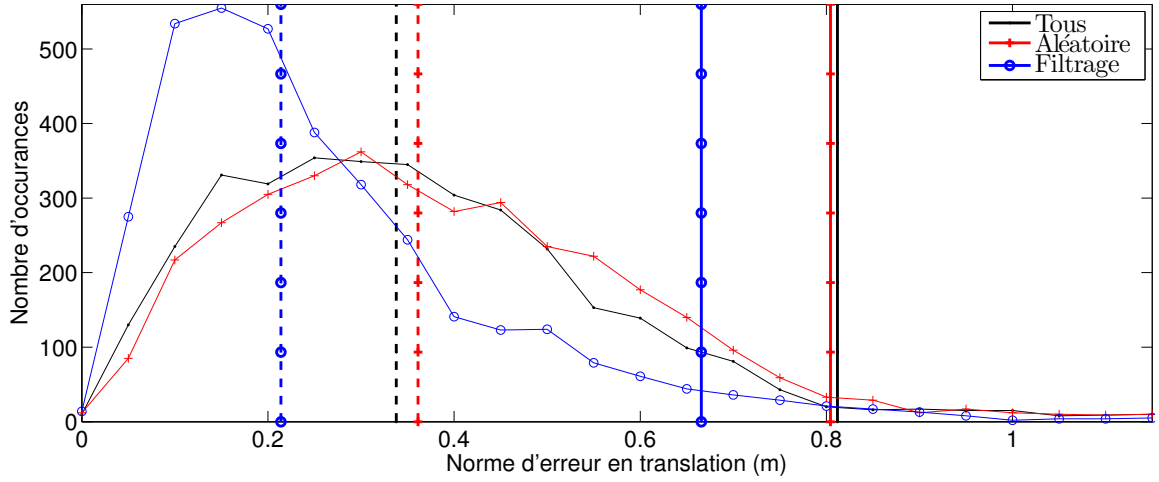


(a) Normes d'erreurs en translation pour la séquence *Wood Summer*.

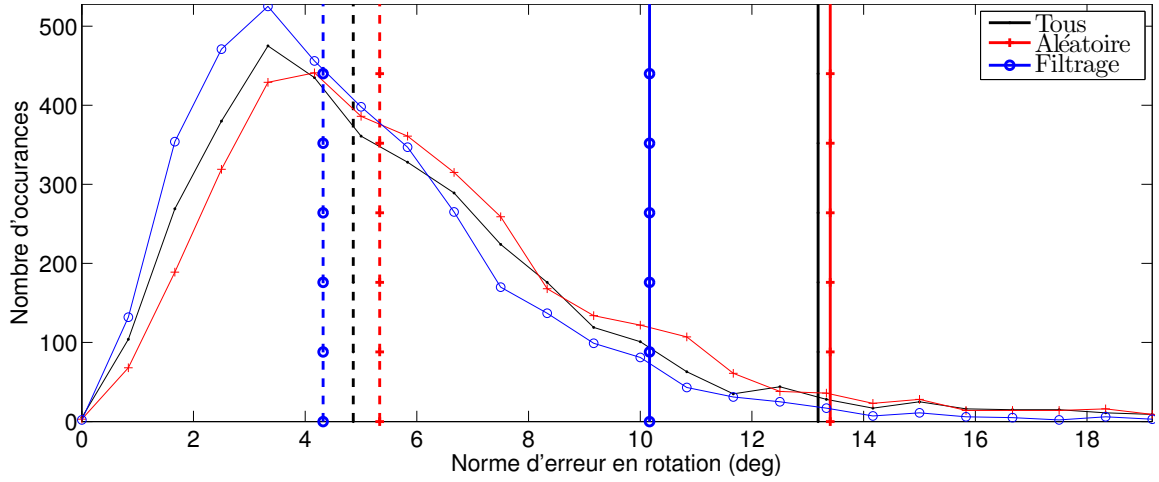


(b) Normes d'erreurs en rotation pour la séquence *Wood Summer*.

FIGURE 5.6: Distributions des normes d'erreurs de calage à travers les scans de la séquence *Wood Summer*, en translation (5.6(a)) et en rotation (5.6(b)). Les résultats ont été groupés par ensembles de six scans consécutifs afin d'alléger les graphiques et de mieux faire ressortir l'évolution générale des performances de chaque méthode à mesure que le robot se déplace.



(a) Distributions des normes d'erreurs en translation pour la séquence *Wood Summer*



(b) Distributions des normes d'erreurs en rotation pour la séquence *Wood Summer*.

FIGURE 5.7: Distributions des erreurs de calage, en translation (5.7(a)) et en rotation (5.7(b)), pour l'ensemble de la séquence *Wood Summer*. Les lignes verticales pointillées indiquent les valeurs médianes des distributions, alors que les lignes verticales pleines indiquent celles des 95^e centiles.

Récapitulatif de l'amélioration du calage par filtrage

Le Tableau 5.1 présente les pourcentages de réduction des erreurs médianes et des 95^e centiles pour les deux séquences de test.

Séquence	Amélioration en translation (%)		Amélioration en rotation (%)	
	Médiane	95^e centile	Médiane	95^e centile
Hannover	17.6	23.6	15.5	17.3
Wood Summer	38.2	17.3	10.3	22.9

TABLE 5.1: Pourcentages de réduction des erreurs médianes et des 95^e centiles pour les séquences de test *Hannover* et *Wood Summer*.

Temps de calcul

L'autre avantage de l'approche proposée est l'accélération significative du processus d'alignement des nuages de points. Le Tableau 5.2 détaille les temps moyens requis pour générer les données d'entraînement, entraîner le classificateur, filtrer les descripteurs et aligner les nuages de points. Les temps de calcul des descripteurs ne sont pas inclus dans le tableau, car ceux-ci sont identiques peu importe la méthode utilisée par la suite.

Une fois le classificateur entraîné, le temps de calcul requis pour caler une paire de scans (Filtrage + Alignement)[3] résulte, pour *Hannover*, en une accélération d'un facteur $17.4\,\mathrm{s}/(0.2\,\mathrm{s} + 2.7\,\mathrm{s}) = 6$ par rapport à lorsque tous les descripteurs sont utilisés. Pour *Wood Summer*, cette accélération est d'un facteur $311\,\mathrm{s}/(2.3\,\mathrm{s} + 15.3\,\mathrm{s}) = 17.7$. En considérant le temps requis pour l'entraînement du classificateur (Gén. + Entraînement), la méthode proposée demeure quand même plus rapide que d'utiliser tous les descripteurs. En effet, ce temps calcul est inférieur au temps pris pour effectuer un seul calage en utilisant tous les descripteurs. Pour *Hannover*, $0.8\,\mathrm{s} + 4.2\,\mathrm{s} + 0.2\,\mathrm{s} + 2.7\,\mathrm{s} = 7.9\,\mathrm{s} < 17.4\,\mathrm{s}$, alors que pour *Wood Summer*, $217\,\mathrm{s} + 29.5\,\mathrm{s} + 2.3\,\mathrm{s} + 15.3\,\mathrm{s} = 264.1\,\mathrm{s} < 311\,\mathrm{s}$. Par conséquent, le processus d'apprentissage est rentable après un seul et unique calage. De plus, la génération des données d'entraînement et l'entraînement du classificateur pouvant être exécutés une seule fois (tant que l'environnement reste similaire), ce temps de calcul est amorti au fil des scans.

5.3 Détails d'implémentation

Pour les résultats des tests présentés ci-haut, tous les calculs ont été effectués séquentiellement, même si toutes les étapes de la méthode proposée peuvent être parallélisées. Les calculs ont été effectués en utilisant les librairies *Point Cloud Library* (PCL) [36] en C++ et Scikit-learn [30] en Python.

3. Le temps de calcul pour le filtrage est compté une seule fois, car l'un des deux scans a normalement déjà été filtré lors du calage précédent.

Séquence	**Tous les descripteurs**	**Approche proposée**			
	Alignement	Gén.	Entraînement	Filtrage	Alignement
Hannover	17.4 s	0.8 s	4.2 s	0.2 s	2.7 s
Wood Summer	311 s	217 s	29.5 s	2.3 s	15.3 s

TABLE 5.2: Temps de calcul moyens (en secondes) pour l'approche proposée et l'approche utilisant tous les descripteurs.

Dans le tableau présentant les temps de calcul moyens, on remarque que la génération des données est très longue pour *Wood Summer*. Cela est dû en partie au nombre de points plus élevé, augmentant le temps nécessaire à la construction du *k-d tree* et à la recherche du plus proche voisin de chaque descripteur. Le processus de recherche des plus proches voisins pourrait être grandement accéléré en utilisant *Fast Library for Approximate Nearest Neighbor* (FLANN) [28]. Brièvement, cette méthode fait des approximations pour trouver le vrai plus proche voisin avec une certaine probabilité, au lieu de le trouver avec certitude (ce qui est fait avec un *k-d tree* standard). Une autre cause de ce temps de calcul élevé est possiblement l'espace mémoire nécessaire pour stocker les valeurs des descripteurs qui est beaucoup plus grand dans le cas de *Wood Summer*, ce qui peut engendrer des ratés de cache plus fréquents. Cette hypothèse n'a toutefois pas été vérifiée.

Chapitre 6

Conclusion et travaux futurs

Dans ce mémoire, nous avons traité du problème du calage de points 3D en robotique mobile. Nous avons d'abord couvert les concepts de calage de nuages de points et de descripteurs, ainsi que quelques notions de base d'apprentissage automatique. Les difficultés engendrées par la présence du feuillage dans les environnements naturels ont ensuite été abordées. Il a été démontré que les descripteurs de points situés dans le feuillage sont instables et, par conséquent, non fiables pour l'estimation de correspondances et le calage. La solution qui a été proposée pour améliorer les performances du calage dans ce type d'environnement consiste à entraîner un classificateur pour filtrer les descripteurs. Nous avons montré que le paradigme *positive and unlabeled* est bien adapté à ce problème d'apprentissage. Nous avons donc adapté l'algorithme *random forest*, modifié dans sa phase de bootstrapping pour rebalancer les classes, à ce paradigme. Cette approche a donné place à un algorithme rapide et robuste qui, entraîné dans un certain environnement, permet de prédire la fiabilité des descripteurs pour l'estimation de correspondances et le calage. Un aspect clé de l'approche proposée est que le robot génère et étiquette les données d'entraînement lui-même, sans aucune intervention humaine requise. Les résultats obtenus montrent que l'ensemble de descripteurs filtrés procure deux avantages à l'algorithme de calage SAC-IA : une meilleure précision et robustesse du calage grossier, ainsi qu'un temps de calcul significativement réduit.

Les travaux que nous avons présentés montrent donc que, en utilisant un scanner laser, la localisation d'un robot peut être améliorée lorsque celui-ci apprend de lui-même avec des données récoltées en cours de route. La publication dans IJCAI-13 de la méthode proposée permet d'exposer la communauté scientifique non seulement à cette nouvelle façon de réaliser le filtrage des nuages de points, mais aussi au paradigme d'apprentissage *positive and unlabeled*, encore peu connu actuellement. Ces travaux constituent donc un pas de plus vers la réalisation d'une navigation autonome robuste d'un robot mobile dans un environnement inconnu.

Par ailleurs, le code et la documentation nécessaires à l'utilisation de la méthode proposée

seront disponibles à la communauté à la date de publication de notre article (août 2013). Ceci facilitera l'intégration de notre solution à un système réel. Une telle intégration pourrait d'ailleurs être testée à l'Université Laval dans le futur.

Même si les résultats obtenus sont encourageants, la méthode proposée assume que les scans utilisés pour l'entraînement sont représentatifs de l'environnement, ce qui peut parfois ne pas être le cas. Également, les résultats sont basés sur l'hypothèse que l'environnement dans lequel le robot évolue reste similaire tout au long de ses déplacements. Par conséquent, une suite intéressante à notre travail serait de permettre au robot d'opérer dans des environnements changeant. Tel que mentionné précédemment, le processus d'entraînement de notre approche peut être répété à la volée, car sa charge computationnelle est relativement faible. Il serait donc possible, en utilisant un mécanisme de détection de changement de l'environnement, de déclencher le processus de réentraînement. Ceci permettrait donc d'adapter le classificateur au nouvel environnement. Un tel mécanisme pourrait consister en un algorithme supervisé de détection d'anomalies [9]. Ce faisant, le déclenchement du réentraînement pourrait s'effectuer en fonction d'un seuil sur le nombre d'anomalies détectées. Une approche plus simple et directe serait de réentraîner le classificateur périodiquement, sans condition, à un intervalle de scans donné.

Enfin, il serait également intéressant de tester notre méthode en utilisant d'autres types de descripteurs, de même que pour des nuages de points acquis avec d'autres types de capteurs, comme la Kinect ou encore des capteurs visuels. Il serait tout aussi intéressant de la tester dans des types d'environnement différents, tels que les terrains planétaires par exemple, qui sont également non structurés.

Bibliographie

[1] Angelova, A., L. Matthies, D. Helmick et P. Perona, « Slip prediction using visual information », *Proc. of robotics : Science and systems*, August 2006.

[2] Belongie, S., J. Malik et J. Puzicha, « Shape matching and object recognition using shape contexts », *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4), p. 509–522, avril 2002.

[3] Besl, P. et H. McKay, « A method for registration of 3-d shapes », *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2), p. 239 –256, feb 1992.

[4] Boser, B. E., I. M. Guyon et V. N. Vapnik, « A training algorithm for optimal margin classifiers », *Proceedings of the fifth annual workshop on computational learning theory*, COLT '92, p. 144–152, New York, NY, USA, ACM, 1992.

[5] Breiman, L., « Bagging predictors », *Machine Learning*, 24, p. 123–140, 1996.

[6] Breiman, L., « Random forests », *Machine learning*, 45(1), p. 5–32, 2001.

[7] Brown, M., G. Hua et S. Winder, « Discriminative learning of local image descriptors », *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1), p. 43 –57, jan. 2011.

[8] *En ligne. Stanford University Computer Graphics Laboratory. The Stanford 3D Scanning Repository*, http ://graphics.stanford.edu/data/3Dscanrep, 2011.

[9] Chandola, V., a. Banerjee et V. Kumar, « Anomaly detection : A survey », 41(3), , p. 15, 2009.

[10] Cortes, C. et V. Vapnik, « Support-vector networks », *Machine learning*, p. 273–297, 1995.

[11] Dahlkamp, H., A. Kaehler, D. Stavens, S. Thrun et G. Bradski, « Self-supervised monocular road detection in desert terrain », *Proc. of robotics : Science and systems*, Aug. 2006.

[12] Efron, B., « Bootstrap Methods : Another Look at the Jackknife », *The Annals of Statistics*, 7(1), p. 1–26, 1979.

[13] Elkan, C. et K. Noto, « Learning classifiers from only positive and unlabeled data », *Proc. of the 14th ACM SIGKDD int'l conf. on Knowledge discovery and data mining - KDD 08*, p. 213, 2008.

[14] Fischler, M. A. et R. C. Bolles, *Readings in computer vision : issues, problems, principles, and paradigms*, chapitre Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography, p. 726–740, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[15] Fontanelli, D., L. Ricciato et S. Soatto, « A Fast RANSAC-Based Registration Algorithm for Accurate Localization in Unknown Environments using LIDAR Measurements », *Ieee international conference on automation science and engineering*, 2007.

[16] Frome, A., D. Huber, R. Kolluri, T. Bülow et J. Malik, « Recognizing objects in range data using regional point descriptors », *European conference on computer vision*, p. 224–237, 2004.

[17] Gagné, C., *Apprentissage et reconnaissance*, Notes de cours, Université Laval, 2011.

[18] Grabner, M., H. Grabner et H. Bischof, « Learning Features for Tracking », *Ieee computer vision and pattern recognition or cvpr*, 2007.

[19] Johnson, A. et M. Hebert, « Using spin images for efficient object recognition in cluttered 3d scenes », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), p. 433 – 449, May 1999.

[20] Klasing, K., D. Althoff, D. Wollherr et M. Buss, « Comparison of surface normal estimation methods for range sensing applications », *International conference on robotics and automation*, p. 3206–3211, 2009.

[21] Krebs, A., C. Pradalier et R. Siegwart, « Adaptive rover behavior based on online empirical evaluation : Rover-terrain interaction and near-to-far learning », *J. of Field Robotics*, 27(2), p. 158–180, 2010.

[22] Latulippe, M., A. Drouin, P. Giguère et F. Laviolette, « Accelerated robust point cloud registration in natural environments through positive and unlabeled learning », *Int'l joint conf. on artificial intelligence (ijcai)*, AAAI Press, 2013. Accepté pour publication.

[23] Lowe, D. G., « Distinctive Image Features from Scale-Invariant Keypoints », *International Journal of Computer Vision*, 60, p. 91–110, 2004.

[24] Mikolajczyk, K. et C. Schmid, « A performance evaluation of local descriptors », *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10), p. 1615–1630, octobre 2005.

[25] Mitra, N. J., N. Gelfand, H. Pottmann et L. Guibas, « Registration of point cloud data from a geometric optimization perspective », *Symposium on geometry processing*, p. 23–31, 2004.

[26] Mordelet, F. et J. Vert, « A bagging svm to learn from positive and unlabeled examples », *arXiv preprint arXiv :1010.0772*, 2010.

[27] Mordelet, F. et J.-P. Vert, *Supervised inference of gene regulatory networks from positive and unlabeled examples*, *Data mining for systems biology*, vol. 939, Methods in molecular biology, p. 47–58, Humana Press, 2013.

[28] Muja, M. et D. G. Lowe, « Fast approximate nearest neighbors with automatic algorithm configuration », *In visapp international conference on computer vision theory and applications*, p. 331–340, 2009.

[29] Ng, A. Y. et M. I. Jordan, « On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. », *Nips*, Dietterich, T. G., S. Becker et Z. Ghahramani, éditeurs, p. 841–848, MIT Press, 2001.

[30] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay, « Scikit-learn : Machine learning in Python », *J. of Machine Learning Research*, 12, p. 2825–2830, 2011.

[31] Pomerleau, F., M. Liu, F. Colas et R. Siegwart, « Challenging data sets for point cloud registration algorithms », *International Journal of Robotics Research*, 31(14), p. 1705 – 1711, December 2012.

[32] Rusinkiewicz, S. et M. Levoy, « Efficient Variants of the ICP Algorithm », *International conference on 3-d imaging and modeling*, p. 145–152, 2001.

[33] Rusu, R., N. Blodow, Z. Marton et M. Beetz, « Aligning point cloud views using persistent feature histograms », *Intelligent robots and systems, 2008. iros 2008. ieee/rsj international conference on*, p. 3384 –3391, sept. 2008.

[34] Rusu, R. B., N. Blodow et M. Beetz, « Fast point feature histograms (fpfh) for 3d registration », *Robotics and automation, 2009. icra '09. ieee international conference on*, p. 3212 –3217, may 2009a.

[35] Rusu, R. B., Z. C. Marton, N. Blodow, A. Holzbach et M. Beetz, « Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments », *The 22nd ieee/rsj international conference on intelligent robots and systems (iros)*, St. Louis, MO, USA, 10/2009 2009b.

[36] Rusu, R. B. et S. Cousins, « 3D is here : Point Cloud Library (PCL) », *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2011.

[37] Rusu, R. B., *Semantic 3d object maps for everyday manipulation in human living environments*, phd, Tecnische Universitatet Muenchen, Munich, Germany, 10/2009 2009.

[38] Sadjadi, F. A. et E. L. Hall, « Three-Dimensional Moment Invariants », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2, p. 127–136, 1980.

[39] Schölkopf, B., J. Platt, J. Shawe-Taylor, A. Smola et R. Williamson, « Estimating the support of a high-dimensional distribution », *Neural computation*, 13(7), p. 1443–1471, 2001.

[40] Scovanner, P., S. Ali et M. Shah, « A 3-dimensional sift descriptor and its application to action recognition », *Proc. of the 15th int'l conf. on multimedia*, p. 357–360, ACM, 2007.

[41] Sim, R. et G. Dudek, « Learning and evaluating visual features for pose estimation », *In proc. of the 7th int'l conf. on computer vision (iccv'99), kerkyra*, p. 1217–1222, IEEE Press, 1999.

[42] Song, M., F. Sun et K. Iagnemma, « Natural feature based localization in forested environments. », *Iros*, p. 3384–3390, IEEE, 2012.

[43] Steder, B., R. Rusu, K. Konolige et W. Burgard, « Point feature extraction on 3d range scans taking into account object boundaries », *Robotics and automation (icra), 2011 ieee international conference on*, p. 2601 –2608, may 2011.

[44] Sun, Y. et M. Abidi, « Surface matching by 3d point's fingerprint », *Eighth international conference on computer vision (iccv'01)*, vol. 2, 2001.

[45] Tangelder, J. W. et R. C. Veltkamp, « A survey of content based 3d shape retrieval methods », *Multimedia Tools Appl.*, 39(3), p. 441–471, septembre 2008.

[46] Tombari, F., S. Salti et L. Di Stefano, « Unique signatures of histograms for local surface description », *Proc. of the 11th european conf. on computer vision : Part iii*, p. 356–369, 2010a.

[47] Tombari, F., S. Salti et L. Di Stefano, « Unique shape context for 3d data description », *Proc. of the acm workshop on 3d object retrieval*, 3DOR '10, p. 57–62, New York, NY, USA, ACM, 2010b.

[48] Trzcinski, T., M. Christoudias, V. Lepetit et P. Fua, *Learning image descriptors with the boosting-trick*, *Advances in neural information processing systems 25*, Bartlett, P., F. Pereira, C. Burges, L. Bottou et K. Weinberger, éditeurs, p. 278–286, 2012.

[49] Winder, S. A. J., G. Hua et M. Brown, « Picking the best daisy », *2009 ieee computer society conf. on computer vision and pattern recognition (cvpr 2009)*, p. 178–185, IEEE, 2009.

[50] Wulf, O., A. Nüchter, J. Hertzberg et B. Wagner, « Benchmarking urban six-degree-of-freedom simultaneous localization and mapping », *J. of Field Robotics*, 25(3), p. 148–163, 2008.