MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

AI-enhanced LoRa Based Indoor Localization System

Fait par
Manon Racine

Sous la direction de
Dr. Nuria Pazos
HE-Arc

St-Imier, HE-Arc, 2 janvier 2019

Abstract

Résumé

Compléter cette partie

Remerciements

Compléter cette partie

Table des matières

Table des figures

1 Introduction

1.1 Contexte

La précision du positionnement intérieur reste quelque chose de très important car il peut être utile dans plusieurs domaines comme : La gestion de stocks, la localisation de personnes âgées dans des homes, la localisation chez eux des personnes possédant un bracelet de "prisonnier", etc...

Les systèmes de localisation basés sur GPS souffrent de la détérioration de la précision et sont presque indisponibles dans les environnements intérieurs. Pour les environnements intérieurs, de nombreuses technologies de systèmes de positionnement ont été conçues sur la base de la vision, de la détection infrarouge ou ultrasonore, des champs magnétiques de la terre, des accéléromètres / gyromètres, des balises BLE ou de la communication WiFi. Bien que la création de ces nouvelles applications ait été couronnée de succès, le coût de ces récepteurs, leur consommation d'énergie et leur limitation aux environnements extérieurs excluent de nombreuses applications.

La Figure 1.1 montre un graphique comparatif de la précision de positionnement concernant différentes technologie [LLI]. Ce graphique montre qu'un système utilisant la vision apporte une grande précision. Mais il peut poser problèmes dans certaine application où il serait nécessaire de mettre beaucoup de caméra et cela implique souvent des traitements assez complexe. En comparaison, un système à ultrason est un peu comparble à un système infrarouge, ils sont meilleurs marché. Il est également nécessaire de placer un grand nombre de capteur selon les utilisations. Concecrnant les systèmes RFID, ils sont de portée très courte qui se limite à quelques mètres et la précision est dégradée par rapport aux systèmes précédents. Les deux derniers systèmes qui sont les réseaux sans fils (bluetooth et WLAN) possèdent beaucoup d'avantage car ils sont largement répendus et possèdent une bonne couverture. Un gros incovénient de ces deux technologies est qu'elles sont gourmandes en énergie et ne peuvent pas fonctionner sur batterie.

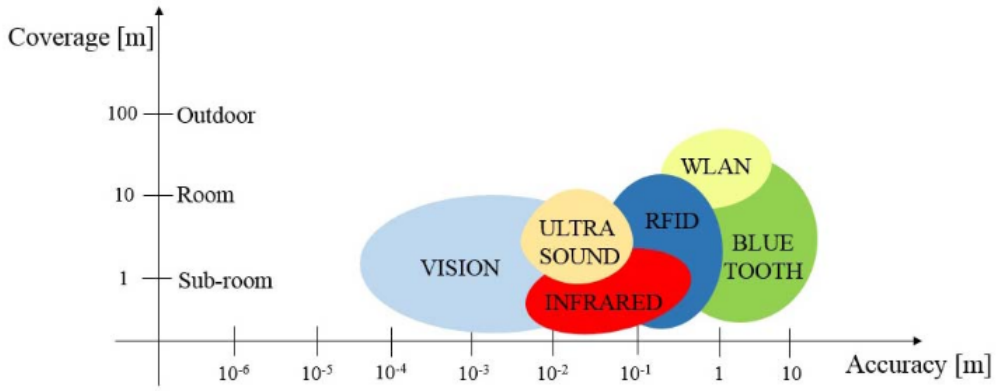


Figure 1.1 – Etat de l'art des différentes méthode de positionnement intérieur [LLI]

La géolocalisation avec LoRa est une possibilité séduisante et probablement l'un des meilleurs candidats pour le positionnement intérieur. Le faible coût des infrastructures et des noeuds finaux ainsi que la disponibilité à l'échelle de la ville ou du pays pourraient permettre de nombreuses nouvelles applications. Il n'est donc pas surprenant que les chercheurs et les entités commerciales se soient mis au travail sur ce problème au cours des derniers mois. Cependant, plusieurs défis restent à relever pour qu'un tel système devienne pratique. Premièrement, la précision de localisation en extérieur qui peut actuellement être atteinte avec LoRa est comprise entre 30 et 50 mètres, ce qui n'est pas suffisant pour de nombreuses applications en milieu urbain. Deuxièmement, LoRa peut également être utilisé pour un positionnement

intérieur. A ce jour, très peu d'expériences sont disponibles pour la conception de tel systèmes et c'est pourquoi il est nécessaire d'effectuer différentes recherche (dont ce travail) afin de connaitre la précision qui peut être atteinte.

La portée en milieu rural est d'environ 15km alors qu'il est de 5km dans un milieu urbain cela grâce à la bonne sensibilité du récepteur (-130dBm). Une chose intéressant est la bande passante qui est plus large que d'autres technologies qui permet de distinguer différents chemins du même signal. Sagemcom ont obtenus des bons résultats au niveau de la précision qui est de environ 4 mètres.[FP17] Un autre aspect très important qui fait de la technologie LoRa est un très bon canditdat pour du positionnement intérieur est sa très faible consommation.

L'objectif général de ce projet est d'étudier diverses technologies d'apprentissage (Machin Learning) permettant d'améliorer la précision de la géolocalisation en intérieur sur la base de la technologie LoRa.

Le positionnement intérieur reste très intéressant et important car il peut être utile dans plusieurs domaines comme : La gestion de stocks, la localisation de personnes âgées dans des homes, la localisation chez eux des personnes possédant un bracelet de "prisonnier", etc...

1.2 Aspect Novateur

Ce travail d'approfondissement va permettre d'évaluer une nouvelle approche pour améliorer le positionnement intérieur. En s'appuyant sur les capacités étendues des nouveaux circuits intégrés LoRa, ce projet développera et déploiera un système de localisation capable d'améliorer la précision de la position atteinte par les systèmes de localisation basés sur LoRa existants reposant sur des mécanismes TDOA ou de "ranging". À cette fin, une exploration et une comparaison des différentes techniques "machine learning/deep learning" pour l'amélioration de la précision du positionnement basé sur le "fingerprinting" seront effectuées.

L'aspect novateur du projet est d'intégrer un mécanisme d'apprentissage de la position afin d'améliorer la précision du positionnement intérieur.

1.3 Structure du rapport

Compléter cette partie en fin de rapport quand la structure est définie

1. Etudier l'état de l'art des techniques à utiliser dans le cadre du projet, en particulier les systèmes de localisation indoor basés sur des techniques d'apprentissage, et réunir une documentation (env. 20
2. Définir un plan des tests à effectuer.
3. Définir les procédures de test
4. Définir le setup pour la collecte de données de localisation
5. Prise en main de l'environnement de développement pour les phases de training et du test de la technique d'apprentissage retenue (e.g., PyTorch).
6. Implémentation de la solution ML retenue.
7. Tester le système selon le protocole préétabli.
8. Faire des propositions pour améliorer les performances de l'algorithme et, si possible, les implémenter.

2 Etat de l'art

Ce chapitre passe en revue les solutions existantes qui parlent d'application des méthodes d'apprentissage pour l'amélioration de la précision de la localisation intérieure. l'exploration de ces sujets va permettre de choisr la solution la plus adaptée pour remplir les objectifs de ce travail.

Pour ce faire, quatre ouvrages ont été étudiés. Les trois premiers traitent du positionnement intérieur aidé avec des algorithmes d'apprentissage (Machine Learning). Le quatrième a été sélectionné car il traite de la gestion de "outliers", c'est à dire comment détecter des points abérant et qui pourraient fausser les mesures.

Grâce à l'étude des structures existantes, une première classification selon le type d'apprentissage et selon les algortheme de classification utilisés peut être établi. En plus, un apperçu de la la façon de détecter les "outliers" est soulevée.

2.1 Type d'apprentissage

Il exite plusieurs catégories d'estimateur et dans ces catégories, il y a plusieurs algorithmes à choix. La Figure 2.1 permet de selectionner différents algorithmes en fonction de ce qu'il est nécessaire d'obtenir. On remarque sur cette figure qu'il y a 4 groupe distincts : classification, regression, clustering et dimensionality reduction. En partant depuis le point "start" et en répondant aux différentes question, cela va proposer un algorithme qui permet de répondre à nos besoins.

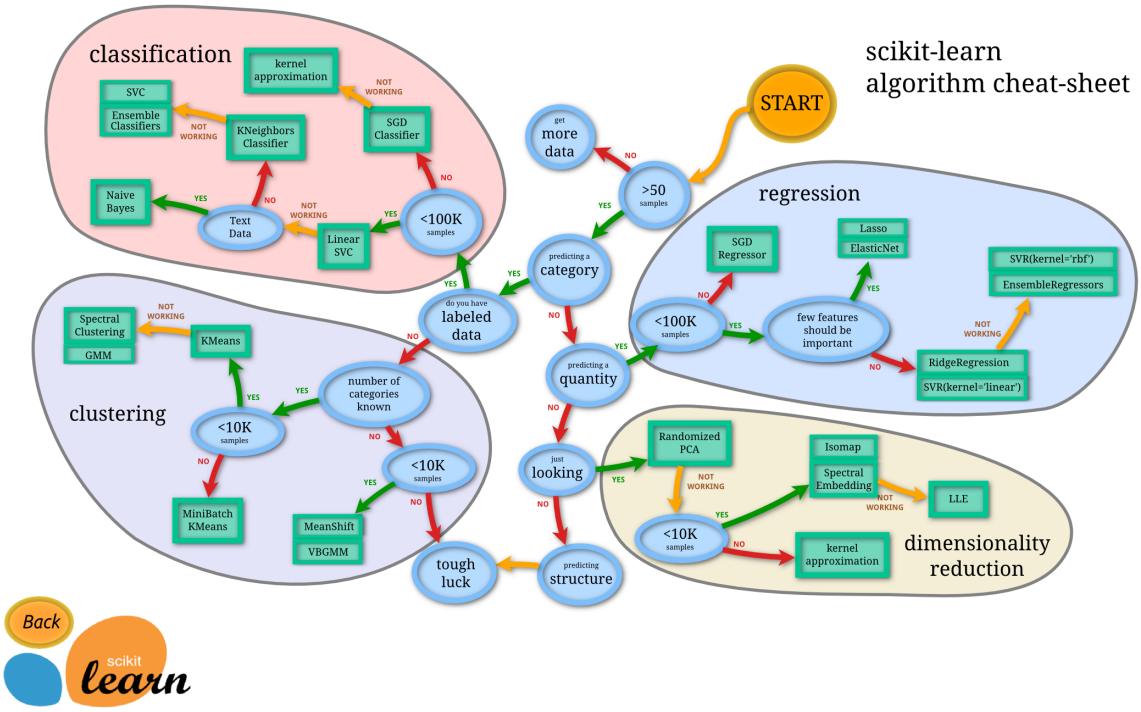


Figure 2.1 – Comment choisir un algorithme ML [Ped+11]

Dans le cas de ce travail, il y a deux possibilités. D'utiliser une classification ou d'utiliser la regression. Dans la Figure 2.2, une petite explication de la différence des deux type d'estimateur. Dans le cas de la classification, on va apprendre à notre algorithme différentes postition et l'algoritme sera ensuite capable de fournir en sortie une de ses positions. Donc, il sort une région plutôt qu'une coordonnée. Si l'entrainement de l'algorithme a été fait avec un maillage fin alors le résultat sera fin. Lorsqu'un point est détecté il sera classifié selon la position la plus proche comme on le voit sur la Figure 2.2 avec le carré vert et le carré rouge. Dans une regression la problématique est un peu différente. L'alorithme sera également entrainé avec différente position mais ensuite, par regression l'algorithme va tenter d'estimer la position du nouveau point en calcuant sa coordonée alors que pour une classification il sagit de positionner ce nouveau point dans une classe.

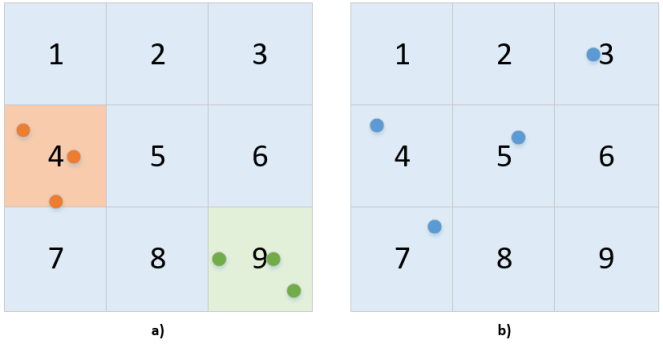


Figure 2.2 – a) exemple de classification b) exemple de regression

2.2 Algorithme de classification

Dans cette section, quelques algorithmes de classification utilisés dans ces études [FP17] [KN17] sont brièvement décrits et analisés.

2.2.1 Decision Tree

L'arbre de décisions et une méthode très connue en "machine learning". Il possède des noeuds de décisions (non-terminal), des branches, et des noeuds feuilles (terminal) qui représentent les caractéristiques, condition et les classes. A chaque noeud de décision on sait quelle branches suivre et lorsque l'algorithme atteint un noeud final, le label contenu dans ce même noeud est retourné comme étant la classe. L'ID3 de Quinlan et son successeur, C4.5, sont les plus populaires parmi les algorithmes d'arbre de décision [JR14].

2.2.2 Naïve Bayes

Le classificateur Naïve Bayes [GHP95] basé sur le théorème de Bayes est un algorithme d'apprentissage supervisé [CS13]. Il est robuste aux données bruyantes, facile à construire, affiche une grande précision et rapidité lorsqu'il est appliqué à de grandes bases de données et exécute des modèles de classification plus complexes. Par conséquent, il est largement utilisé dans les tâches de classification. Il calcule la probabilité de chaque attribut dans les données en supposant qu'elles sont également importantes et indépendantes les unes des autres. Cette hypothèse est appelée indépendance conditionnelle de classe [WA13] [SS14].

2.2.3 Bayesian Network

L'algorithme de réseau bayésien est largement utilisé pour la classification et est basé sur le théorème de Bayes où la probabilité conditionnelle sur chaque nœud est calculée et forme un réseau bayésien. Il

s'appelle également réseau de croyance ou réseau occasionnel. Réseau bayésien a deux parties nommées qualitatives et quantitatives, qui sont la structure topologique du réseau bayésien et le tableau de probabilité conditionnelle (CPT), respectivement [DL07].

Le réseau bayésien est un graphe acyclique dirigé où chaque nœud représente un attribut des données et un ensemble de distributions de probabilité. Ces distributions donnent les probabilités pour la valeur de chaque nœud étant donné que les parents de nœud.

2.2.4 K-Nearest Neighbor

Le classificateur K-Nearest Neighbor (K-NN) [DWDMK91] est également connu sous le nom de classificateur basé sur la distance qui classe les instances en fonction de leur similarité. C'est l'un des algorithmes les plus populaires de l'apprentissage automatique. C'est un type d'apprentissage paresseux dans lequel la fonction n'est approchée que localement et tout calcul est retardé jusqu'à la classification. Le tuple inconnu dans K-NN est assigné à la classe la plus commune parmi ses K-plus proches voisins. Lorsque K = 1, le tuple inconnu se voit attribuer la classe du tuple d'apprentissage le plus proche dans l'espace des motifs [CAG13].

2.2.5 SMO / SVM

L'algorithme d'optimisation séquentielle minimale (SMO - Sequential minimal optimization) [J.98] est représenté par John C. Platt pour la formation du classificateur de vecteurs de support à l'aide des noyaux polynomiaux ou RBF. C'est l'un des algorithmes les plus courants pour la classification des grandes marges par SVM. Il remplace globalement toutes les valeurs manquantes et transforme les attributs nominaux en attributs binaires. La SVM est une technique de classification basée sur la technologie des réseaux neuronaux utilisant la théorie de l'apprentissage statistique [PH14]. Il recherche un hyperplan optimal linéaire afin de maximiser la marge de séparation entre la classe positive et la classe négative. En pratique, la plupart des données ne sont pas linéairement séparables ; ainsi, pour rendre la séparation possible, la transformation est effectuée à l'aide d'une fonction du noyau. L'entrée est transformée en un espace caractéristique de dimension supérieure à l'aide d'une cartographie non linéaire [30]. Une décision sur la fonction du Kernel est nécessaire pour implémenter SVM. Le Kernel définit la classe de fonction [SMKN13].

2.2.6 AdaBoost

AdaBoost (Adaptive Boosting) [YRE96] est un algorithme d'apprentissage d'ensemble. Généralement, il peut être utilisé avec des algorithmes de Machine learning faibles pour améliorer leurs performances. Il est simple à mettre en œuvre, rapide et moins susceptible d'avoir un overfitting. Il améliore les algorithmes de classification instables tels que J48, DecisionStump, etc. L'idée derrière cet algorithme est d'obtenir un classificateur très précis en combinant de nombreux classificateurs faibles. Il fonctionne en exécutant de manière répétée un algorithme d'apprentissage faible donné sur diverses distributions sur les données d'apprentissage, puis en combinant les classificateurs produits par l'apprenant faible en un classificateur composite unique [RRE13]. Les classificateurs de l'ensemble sont ajoutés un par un, de sorte que chaque classificateur suivant est entrainé sur des données difficiles pour les membres précédents de l'ensemble. Les poids sont définis sur les instances du jeu de données, en suivant une règle selon laquelle les instances difficiles à classer prennent plus de poids. Cette règle conduit les classificateurs ultérieurs à se concentrer sur eux [SOLISP10].

2.2.7 Bagging

Le Bagging [L.96] crée des sacs de données de la même taille que le jeu de données d'origine en appliquant une sélection aléatoire à différents sous-ensembles des données d'apprentissage avec de nombreux exemples qui apparaissent plusieurs fois. Ce processus est appelé réplication bootstrap des données d'entrainement. L'idée derrière cette technique est de construire différents classificateurs en utilisant ces sous-ensembles.

Chaque sous-ensemble est utilisé pour entrainer un classificateur individuel. Cette approche d'ensemble utilise le nombre de classificateurs a priori [L.96].

2.3 Algorithme de regression

Dans cette section, quelques algorithmes de regression utilisés dans cette étude [Hen+12] sont brièvement décrits et analysés.

2.3.1 Support vector machine pour la régression (SVR)

Support vector machines (SVMs) est une méthode d'apprentissage suppervisée utilisée pour la classification. Cette méthode peut être étendue pour résoudre des problèmes de régression et cette méthode s'appelle Support Vetor Regression (SVR)

Systeme Vector Regressio (SVR) est considéré comme une technique non paramétrique car il dépend de fonction du kernel.

Il existe trois implémentations différentes de Support Vector Regression : SVR, NuSVR et LinearSVR. LinearSVR fournit une implémentation plus rapide que SVR mais ne considère que les noyaux linéaires, tandis que NuSVR implémente une formulation légèrement différente de SVR et de LinearSVR.[Ped+11]

2.3.2 Gaussian Process (GP)

Gaussian processes a récemment gagné en interet du coté des algorithme d'apprentissage. Cela car il forme un bon framework pour résoudre des problèmes de régression [CC06].

C'est une méthode d'apprentissage supervisée pour résoudre des problèmes de regression comme dit ci-dessus mais également de porblème de probabilité.

Les avantage du processus gaussien sont :

1. La prédiction interpole les observations (au moins pour les noyaux normaux).
2. La prédiction est probabiliste (gaussienne) afin que l'on puisse calculer des intervalles de confiance empiriques et décider en fonction de ceux-ci s'il convient de réajuster (adaptation en ligne, adaptation adaptative) la prévision dans une région d'intérêt.
3. Polyvalent : différents noyaux peuvent être spécifiés. Les noyaux communs sont fournis, mais il est également possible de spécifier des noyaux personnalisés.

Les inconvénients des processus gaussiens sont :

1. Ils ne sont pas clairsemé, c'est-à-dire qu'ils utilisent l'ensemble des informations sur les échantillons / caractéristiques pour effectuer la prédiction.
2. Ils perdent leur efficacité dans les espaces de grandes dimensions, notamment lorsque le nombre de caractéristiques dépasse quelques dizaines.

[Ped+11].

2.4 Détection d'outliers [FP17]

Cette section donne un apperçu de la manière de traiter les "outliers - valeurs aberrantes", c'est-à-dire les points qui ne sont pas cohérent lors d'une mesure. Selon Barnet et Lewis [VT94], un "outliers" est définit comme étant une observation qui semble incompatible avec le reste d'un ensemble de données. Garder un "outliers" dans une set de données peut amener à de mauvais résultats, il est donc important de les détecter correctement. Il existe différentes méthodes pour déterminer ces "outliers" :

1. Grubbs' test : Détecte un "outliers" en supposant une distribution normale.
2. Tietjen-Moore test : C'est une généralisation de Grubbs' test pour détecter de multiple outliers. Il a cependant un inconvénient, il est nécessaire de connaitre le nombre exact d'ouliers.
3. Generalized Extreme Studentized Deviate (ESD) : C'est également une généralisation du test Grubbs' mais il n'est pas nécessaire de connaitre à l'avance le nombre d'ouliers. Ce test nécessite uniquement une limite supérieure pour le nombre suspect d'outliers.[Esd]

2.5 Etude des travaux existants

Les trois documents ne peuvent pas être comparé à proprement dit car ils traitent de différentes manières. Un parle d'algorithme de classification et compare différents algorithmes existants. Un autre document traite de régression et compare également différents algorithmes. Le dernier document parle de la prédiction conforme (CP - conformal prediction) qui va permettre d'améliorer encore les résultats avec un algorithme choisi.

C'est pourquoi les documents sont résumés en mettant les points importants en évidence séparément sera décrit plutôt qu'une comparaison.

2.5.1 Analyse comparative de différents algorithme pour le positionnement intérieur [Sin+15]

Cette publication parle d'une analyse comparative entre différents algorithmes de "machine learning" pour du positionnement indoor. L'étude est basée sur un positionnement "fingerprint" ce qui permet de cartographier un endroit à l'aide de la force du signal réceptionné (RSS - Received Signal Strength). Dans cet article, les algorithmes de "machine learning" sélectionnés sont comparés en termes de précision de positionnement et de temps de calcul.

Au cours des expériences, la base de données UJIIndoorLoc est utilisée. La classification est effectuée en premier lieu en utilisant le jeu de données d'origine en considérant les valeurs RSS de 520 points d'accès sans fil (WAP - wireless access points) et les nouveaux attributs définis en tant que «cellule» qui composent les attributs BuildingID, Floor, SpaceID et RelativePos. Ensuite, une nouvelle méthode est proposée : «Séparation déductive pour le positionnement intérieur (DESIP - Deductive Separation for Indoor Positioning)». Dans cette méthode, tout d'abord, seules les informations de bâtiment et les valeurs RSS mesurées à partir de 520 WAP sont utilisées pour la tâche de classification.

Durant les expériences, des algorithmes tels que le plus proche voisin (NN - nearest neighbor), le SMO, l'arbre de décision (J48), Naïve Bayes et Bayes Net sont utilisés. L'algorithme le plus approprié pour la solution du problème de positionnement intérieur est déterminé en comparant la précision et le temps de calcul de chaque approche.

La base de données entière est séparée de telle sorte que 19.937 enregistrements soient réservés à l'apprentissage et 1.111 enregistrements soient réservés aux tests. Il y a 529 caractéristiques et ces caractéristiques sont les coordonnées où sont prises les empreintes digitales WiFi, telles que bâtiment, étage, espace (bureau, laboratoire, etc.), position relative (dans une pièce ou dans un couloir), etc. Le jeu de données d'apprentissage UJIIndoorLoc comprenant les valeurs RSS de 520 WAP et un nouvel attribut «cellule» qui compose les attributs floor, buildingID, spaceID et relative position de l'ensemble de données d'origine est utilisé pour la tâche de classification. Les étapes des expériences utilisant ce jeu de données sont illustrées à la Figure 2.3. Dans un premier temps, le dataset est séparé en deux groupes. Un groupe est utilisé pour

l'apprentissage (train dataset) et l'autre partie est utilisée pour valider le résultat des différents algrotihme (test dataset). Le dataset d'entrainement est utilisé pour quatre différents algorithmes.

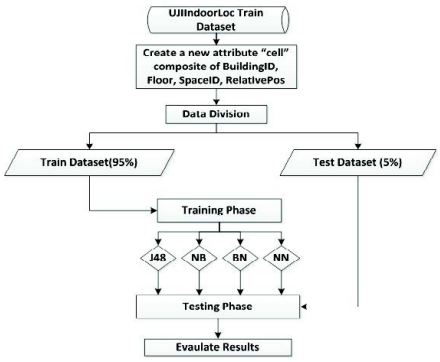


Figure 2.3 – Phase de construction et d'apprentissage [Sin+15]

Dans cet article les algorithmes suivants ont été comparés dans différentes situation : NN, SMO, J48, Naïve Bayes and BayesNet. La classification est effectuée en 3 phases (building,floor and region). La première étape est la classification par bâtiment (Figure 2.4). Le résultat des algorithmes pour cette classification donne BayesNet comme étant le meilleur car il possède la meilleure précision.

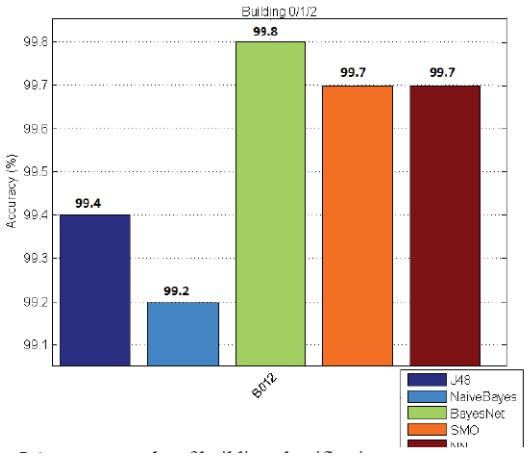


Figure 2.4 – Résultats des précisons pour la classification par bâtiments [Sin+15]

Suite à cela, la classification a été faite en fonction des étages (floors) (Figure 2.5). Dans ce cas, le meilleur algorithme est NN.

Et pour la dernière étape (Figure 2.6), la classification a été faite en fonction de la région et là encore, c'est l'algorithme NN qui est le meilleur.

Les deux tableaux (Figure 2.7 et Figure 2.8) permettent d'avoir une vue d'ensemble de tous les algorithmes appliqués et leur performance en terme de précision et de rapidité. L'algorithme NN est le meilleur pour tous les dataset niveau du temps d'exécution. En ce qui concerne la précision, Bayes Net est meilleur pour la classification "building" par contre NN est meilleur dans tous les autres cas.

Les résultats expérimentaux révèlent que l'algorithme k-Nearest Neighbor (k-NN) est le plus approprié lors du positionnement. En outre, J48 offre des performances quasiment identiques lorsqu'il est utilisé avec des algorithmes itératifs, à savoir AdaBoost et Bagging. Il est à noter que toute ces méthodes mettent en évidence des classifications. C'est à dire que le résultat placera un nouveau point sur l'un des points entrainé et ne donnent pas une nouvelle coordonnée comme cela serait le cas d'un résultat par regression.

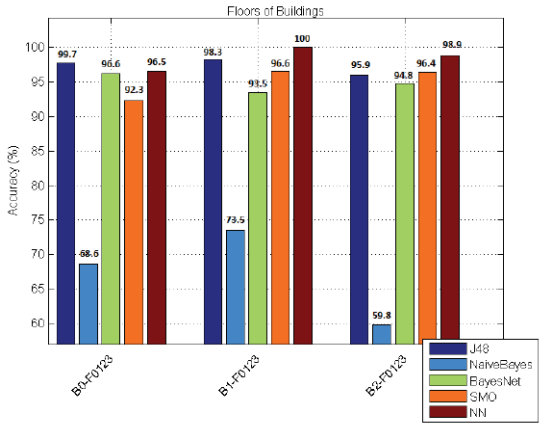


Figure 2.5 – Résultats de la précision pour la classification par étage.[Sin+15]

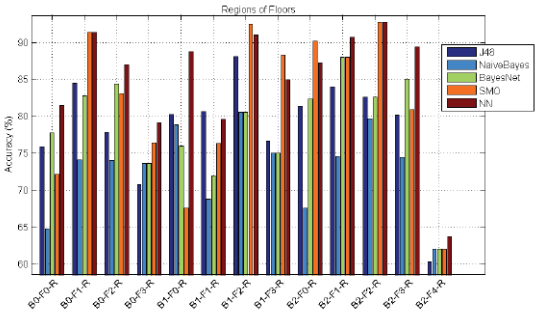


Figure 2.6 – Résultats de la précision pour la classification par région. [Sin+15]

	J48	Naive Bayes	Bayes Net	SMO	NN
B012	99,4	99,19	99,80	99,69	99,69
B0_F0123	97,7	68,58	96,17	92,34	96,55
B0_F0_R	75,93	64,81	77,78	72,22	81,48
B0_F1_R	84,48	74,14	82,76	91,37	91,38
B0_F2_R	77,92	74,03	84,42	83,12	87,01
B0_F3_R	70,83	73,61	73,61	76,39	79,17
B1_F0123	98,28	73,54	93,47	96,56	100
B1_F0_R	80,28	78,87	76,06	67,61	88,73
B1_F1_R	80,64	68,82	72,04	76,34	79,57
B1_F2_R	88,06	80,60	80,59	92,54	91,04
B1_F3_R	76,67	75	75	88,33	85
B2_F01234	95,95	59,77	94,83	96,40	98,87
B2_F0_R	81,37	67,65	82,35	90,19	87,25
B2_F1_R	84	74,67	88	88	90,67
B2_F2_R	82,61	79,71	82,61	92,75	92,75

Figure 2.7 – Résultat de la précision des algorithmes d'apprentissage.[Sin+15]

	J48	Naive Bayes	Bayes Net	SMO	NN
B012	62,41	2,5	11,19	7,25	0,06
B0_F0123	11,74	0,65	2,03	14,31	0,01
B0_F0_R	1,5	0,22	0,7	14,1	0
B0_F1_R	1,71	0,21	0,88	17,93	0
B0_F2_R	2,52	0,2	0,9	18,77	0
B0_F3_R	1,62	0,2	0,88	17,3	0
B1_F0123	10,05	0,57	2,13	11,5	0,02
B1_F0_R	1,68	0,18	0,87	18,22	0
B1_F1_R	1,75	0,21	1,01	13,3	0
B1_F2_R	1,96	0,27	1,11	19,47	0
B1_F3_R	0,95	0,13	1,11	12,1	0
B2_F01234	18,09	1,18	4,4	18,81	0,03
B2_F0_R	2,54	0,24	1,11	21,57	0
B2_F1_R	3,27	0,29	1,15	27,1	0
B2_F2_R	1,86	0,2	1,46	19,76	0

Figure 2.8 – Résultats du temps de traitement des algorithmes d'apprentissage[Sin+15]

2.5.2 Machine learning pour diminuer l'erreur de positionnement UWB [Hen+12]

Les approches classiques pour faire face au défi de localisation dans des environnements encombrés impliquent généralement d'abord la détection de la condition NLOS, puis la prise de mesures appropriées pour prendre en compte la condition NLOS. Toutefois, la grande variété de matériaux et d'environnements d'exploitation variés peuvent impacter les performances de la mesure de ranging, ce qui indique que la distinction entre LOS et NLOS n'est pas toujours significative. Sur la base de cette observation, ils ont adopté pour une approche différente dans cet article. Leur approche utilise des techniques de Machine Learning non paramétriques (SVM et GP) pour estimer l'erreur de "ranging" directement à partir de la forme d'onde reçue, sans aucune connaissance a priori ou à posteriori de la condition NLOS.

Cette publication traite de la diminution de l'erreur du mode ranging concernant la localisation UWB (Ultra Wide Band). Plusieurs techniques existent pour diminuer l'erreur de positionnement en détectant ce qui est en ligne de vue (LOS) ou non (NLOS). Ici, une autre technique est exploité et qui va directement diminuer cet erreur que ça soit en LOS ou NLOS. Ils appliquent deux classes de régresseurs non paramétriques pour avoir une estimation de l'erreur de mesure. Afin de valider leurs résultats, ils ont fait un vaste campagne de mesures intérieures. Cette technique montre une amélioration de performances significatives dans divers scénarios par rapport aux approches conventionnelles.

La régression non paramétrique est une forme d'analyse de la régression dans lequel le prédicteur, ou fonction d'estimation, ne prend pas de forme prédéterminée, mais est construit selon les informations provenant des données. La régression non paramétrique exige des tailles d'échantillons plus importantes que celles de la régression basée sur des modèles paramétriques parce que les données doivent fournir la structure du modèle ainsi que les estimations du modèle. [Wika]

Ils se sont appuyé sur des outils de Machine Learning, et proposent deux techniques de régression non paramétriques pour estimer l'erreur de mesure, en se basant uniquement sur la forme d'onde reçue et la distance estimée.

1. La première technique utilise une régression SVM (support vector machine) pour trouver un hyperplan qui se rapproche de l'erreur de mesure en fonction des données d'apprentissage.
2. La seconde technique utilise un processus gaussien (GP) pour déterminer la distribution à posteriori de l'erreur de mesure, en fonction des données d'apprentissage.

L'erreur de mesure estimée, associée à une mesure de certitude, peut être transmise à un algorithme de localisation. Leurs techniques de régression présentent l'avantage supplémentaire de pouvoir être appliquées même lorsque les données d'apprentissage ne sont pas étiquetées avec des informations LOS ou NLOS.

Lors des mesures, il existe beaucoup de paramètres qui peuvent créer une erreur. Avec un seul modèle, il est difficile de capturer tout les types de perturbations. Dans cette publication, ils se basent sur 1024 mesures (512 LOS et 512 NLOS) Figure 2.9.

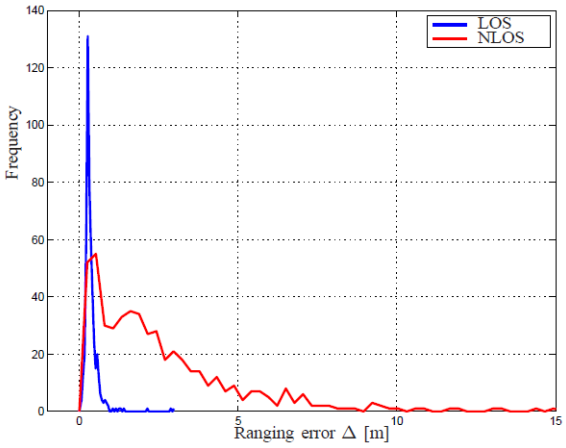


Figure 1. Histogram of the ranging error for the LOS and NLOS condition.

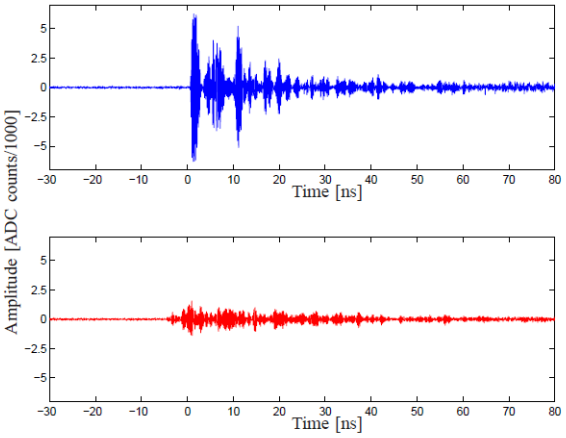


Figure 2. In some situations there is a clear difference between LOS (upper waveform) and NLOS (lower waveform) signals.

Figure 2.9 – Ranging error - LOS and NLOS [Hen+12]

Sur la base d'une vaste campagne de mesures en intérieur avec des radios UWB conformes à la FCC (Federal Communications Commission), ils ont évalué les performances de localisation en termes de probabilité de panne pour différentes stratégies de localisation. Leurs résultats ont révélé que :

1. La minimisation de l1-norme est plus robuste pour faire face aux valeurs aberrantes (outliers) que la minimisation de l2-norme, pour une localisation sans atténuation.
2. les contraintes peuvent générer des gains significatifs, en particulier lorsque les exigences de localisation ne sont pas trop strictes.
3. les techniques de régression SVM ou GP offrent des gains de performance supplémentaires pour tous les scénarios considérés.
4. Les techniques de régression SVM ou GP, combinées à la connaissance des contraintes relatives à l'erreur de "ranging", offrent les meilleures performances pour les scénarios considérés.

2.5.3 Amélioration du positionnement en utilisant "conformal prediction" [KN17]

Pareil que dans les deux précédentes publications, le but est d'améliorer la précision du positionnement intérieur où il n'est pas possible d'utiliser un GPS. Cela toujours en tenant compte des problématiques d'un environnement dynamique avec des personnes qui bougent et de l'environnement complexe avec des murs, etc.. Dans cet article, ils ont validé leur solution dans 3 immeubles. Cet article est basé sur un positionnement WIFI fingerprinting et se base sur la force du signal reçu.

Pour effectuer les mesures il y a deux phases voir Figure 2.10.

La première phase (Off-line training) consiste à décider combien de mesures on veut faire (tous les mètres ? centimètres ?), comment sont prises les mesures (plusieurs mesures sont faites à chaque position) et comment labéliser le signal (souvent labélisé avec la coordonnée réelle). Cette phase est très importante et il est nécessaire de bien réfléchir comment procéder (utiliser un robot par exemple). C'est ce qu'on appelle la phase d'entrainement.

Concernant la seconde phase (on-line positioning) c'est de définir le positionnement. Durant cette phase la partie délicate est de définir quel algorithme utiliser. C'est durant cette phase qu'on va tester les algorithme.

Il existe une compétition comparative pour les positionnements indoor (Microsoft IPSN 2014) Figure 2.11.

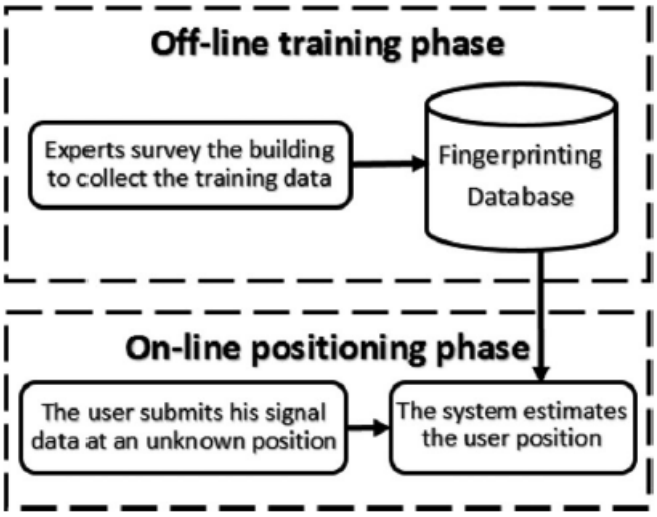


Figure 2.10 – Les deux phases du fingerprinting

Une localisation précise à l'intérieur peut potentiellement transformer la façon dont les gens naviguent à l'intérieur, de la même manière que le GPS a transformé la façon dont les gens naviguent à l'extérieur. Au cours des 15 dernières années, le monde universitaire et l'industrie ont proposé et expérimenté plusieurs technologies de localisation en intérieur, mais nous n'avons pas encore assisté à des déploiements à grande échelle. Ce concours vise à rassembler des technologies de localisation intérieure en temps réel ou quasi réel et à comparer leurs performances dans le même espace. [Mic]

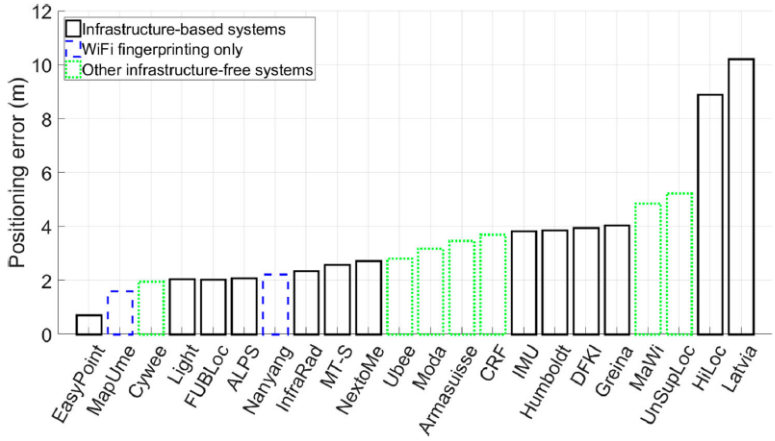


Figure 2.11 – Performance de la précision du fingerprint lors de la compétition Microsoft IPSN 2014 competition [KN17]

Cela permet de mettre en évidence les précisions qui sont obtenues. Les algorithmes qui seront étudiés sont : Weighted K-nearest neighbours (W-KNN), Naïve Bayes Figure 2.12. Tous les mesures et tests sont basé sur le RSS du WIFI.

Les mesures ont été effectuées de trois manières différentes (dans un corridor, sur un étage, sur trois étages - voir dans le document [KN17]). En résumé, la synthèse des trois solutions suggèrent que, avec uniquement la mesure métrique WiFi RSS, de nombreux algorithmes complexes risquent de ne pas être aussi performants que des algorithmes plus simples. Malgré sa simplicité, W-KNN a excellé dans la plupart des analyses de "fingerprinting". Il convient de noter que le système MapUme, deuxième sur 22 concurrents du concours Microsoft IPSN 2014, utilisait également W-KNN comme principal algorithme. Cependant, l'approche Naïve Bayes améliore sa précision lorsque le nombre d'entrainements est élevé, ce qui indique

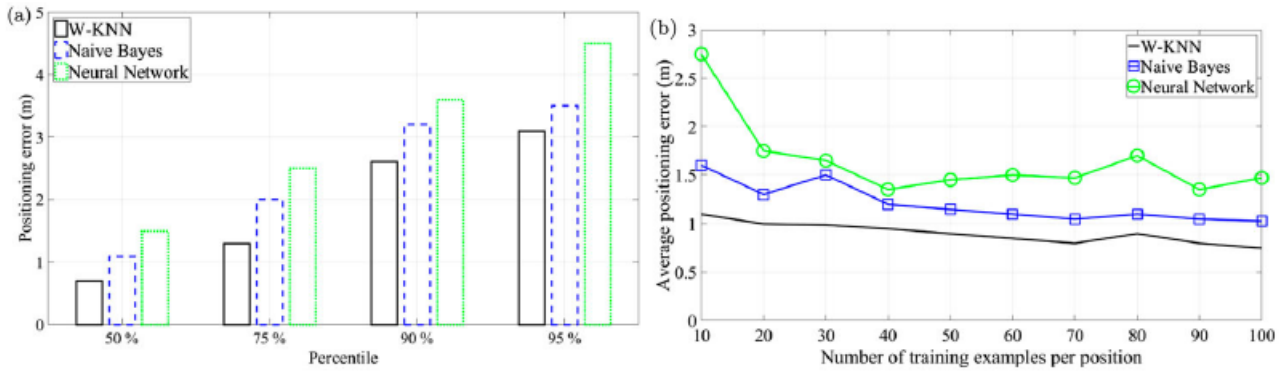


Figure 2.12 – Performance de la précison de W-KNN, Naïve Bayes et réseau de neuronne mais aussi l'impacte du nombre de données d'entrainement[KN17]

qu'au-delà du WiFi RSS, des informations supplémentaires seront nécessaires pour améliorer d'avantage les performances du "fingerprinting".

Une chose qui est également importante c'est la confiance qu'il y a dans un algorithme. Pour cela cet article à utilisé "conformal prediction (CP)" afin de donner un indice de confiance. Plus on souhaite de confiance plus il est nécessaire d'avoir de set de données. Les avantages d'utiliser l'indice de confiance sont les suivants :

1. Chaque prédiction est associée à un indice de confiance et permet de dire combien la prédiction est correcte.
2. La prédiction produite par CP est statistiquement correcte sous les paramètres qui ont été choisi dans la phase on-line.
3. le niveau de confiance peut être ajusté pour produire un ensemble de prédiction plus grand ou plus petit.

Afin de valider leur algorithme, ils ont utilisé 3 bancs de test Figure 2.13.

1. Royal Holloway : Données récoltées manuellement dans un office standard avec un smartphone
2. Cambridge : Données récoltées automatiquement par un robot dans un environnement assez idéal.
3. UJIIndoorLoc : Utilise le dataset publique qui couvre une grande surface indoor basé sur trois bâtiments.

Pour résumer cet article, il propose une nouvelle approche d'apprentissage basé sur la confiance d'un algorithme permettant d'estimer la position de l'utilisateur à l'intérieur avec la force du signal WiFi. Il introduit une mesure de confiance, non seulement utile pour refléter l'incertitude des prédictions de positionnement, mais également capable d'ajuster la taille de l'ensemble de prédictions en conséquence.

Il a été montré empiriquement que la précision de positionnement était d'environ 2,4 m / probabilité de 75% avec un banc d'essai normal, environ 70 cm / probabilité de 75% avec un banc d'essai idéal et d'environ 8,8 m / probabilité de 75% avec un banc d'essai difficile.

Ces résultats ont surpassé les algorithmes de Machine Learning sans indice de confiance mis à l'essai sur les mêmes bancs d'essai jusqu'à 20% plus précis. Si on ne tient pas compte de CP l'algo W-KNN est un peu meilleur que Naïve Bayes.

Les approches présentée dans cet article ne nécessite pas de carte du bâtiment. Cela pourrait être utile de les avoir afin d'avoir des informations supplémentaires pour supprimer les mauvaises prédictions comme par exemple une personne qui marche dans un mur alors qu'elle est dans le couloir.

Table 4. Summary of the three fingerprinting test beds used in this article.

	Royal Holloway	Cambridge	UJIIndoorLoc
Abbreviation	RH	Cam	UJI
Training examples	13,600	78,000	19,937
Training area	1480 m^2	540 m^2	110,000 m^2
Surveyed space	3 corridors	1 corridor, 1 room	3 buildings
Training time	1 day	2 days	20 days
Granularity	1 m	10 cm for room 30 cm for corridor	Up to 2 m
Test samples	150	100	1,111
Last test sample	Same day	1 day later	3 months later
Measuring time	Working hours	Weekend	Working hours
Measures per location	200	40	Up to 30
Distinct positions	68	1950	933
Orientations per location	4	4	Unknown
Label type	Cartesian (metre scale)	Cartesian (metre scale)	Longitude and latitude
Label generator	Manually by the surveyor	Automatically by Active Bat system	Manually by the surveyor
Total building(s)	1	1	3
Total floor(s)	1	1	13
WiFi APs	131	43	529
Device(s) used	1 phone	1 netbook	25 phones
Surveyor(s)	1 person	1 robot	18 people

Figure 2.13 – Summary of the three fingerprinting test beds used in this article [KN17]

2.6 Choix et conclusion

Le choix de l'algrotithme suite à l'analyse de l'état de l'art n'est pas chose aisée. Il est important de garder en tête l'application final pour laquelle le système sera utilisé, c'est-à-dire, pour faire du positionnement intérieur.

En premier lieu, il est nécessaire de choisir entre un apprentissage par classification ou par regression. Il est clair que pour une utilisation finale un algorithme traitant de la regression commme SVM serait le mieux adapté. Ceci afin d'entrainer le système avec peu de position et d'estimer les autres positions.

Cependant, il n'est pas vraiment réaliste de démarrer directement avec une technique pareil sans même savoir si les données disponible sont exploitables. C'est pourquoi, la décison a été de démarrer avec un algoritme de classification dans l'optique d'améliorer le système avec un algorithme de regression.

Suite, à l'étude des documents existants, il aurait été judicieux de sélectionner l'algorithme KNN qui a donnée dans la plus part des cas les meilleurs résultats. Mon choix c'est tout de même tourné vers l'algorithme SVM qui est également bien classé et surtout car il est également cité pour la regression en tant que SVR.

3 Prise de mesures

Ce Chapitre traite de la collecte des données qui est un point important de ce travail. Il a été nécessaire de réfléchir quoi prendre et de quel manière. Pour ce faire, Michael Muller qui avait réalisé sa thèse de master concernant le positionnement indoor a réalisé un programme python fonctionnant sur PC et permettant de récupérer les mesures [Mic17]. Ci-dessous, ce programme sera détaillé plus précisément afin de pouvoir le prendre en main et réaliser les prises de mesures.

Ci-dessous, il sera également précisé comment les données on été structurée afin de pouvoir être utilisée plus tard dans un algorithme de machine learning.

Finalement il sera détaillé comment les mesures ont été effectuées selon le plan du laboratoire.

3.1 Setup pour la prise de mesures

Pour effectuer les mesures, il est nécessaire d'effectuer un certain nombre de mesure afin d'avoir un nombre acceptable de données. L'idée est de respecter le plan ci-dessous voir la figure 3.1. Dans un premier temps, les mesures seront effectuée dans une seule moitié du laboratoire.

Le point rond "M" représente le master, les points ronds "S" représentent les slaves et finalement les points carrés "E" représentent les espions. C'est sur ces derniers que les mesures seront effectuées.

Il sera nécessaire de prendre 20 mesures sur chaque point. Comme uniquement la première moitié sera considéré les mesures seront faites sur 10 points différents donc 200 données seront à disposition. Une mesure consiste à changer de canal de 1 à 40.

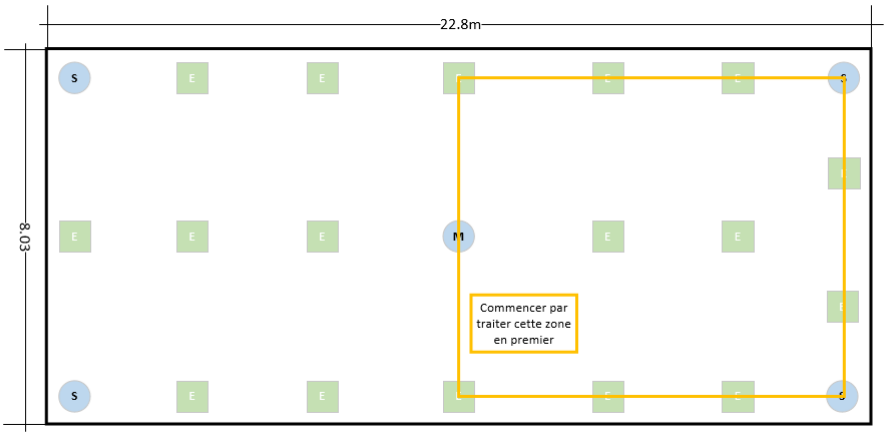


Figure 3.1 – Montre le setup pour la prise de mesure

Le plan de la Figure 3.1 et un simplification du plan réel de la Figure 3.2.

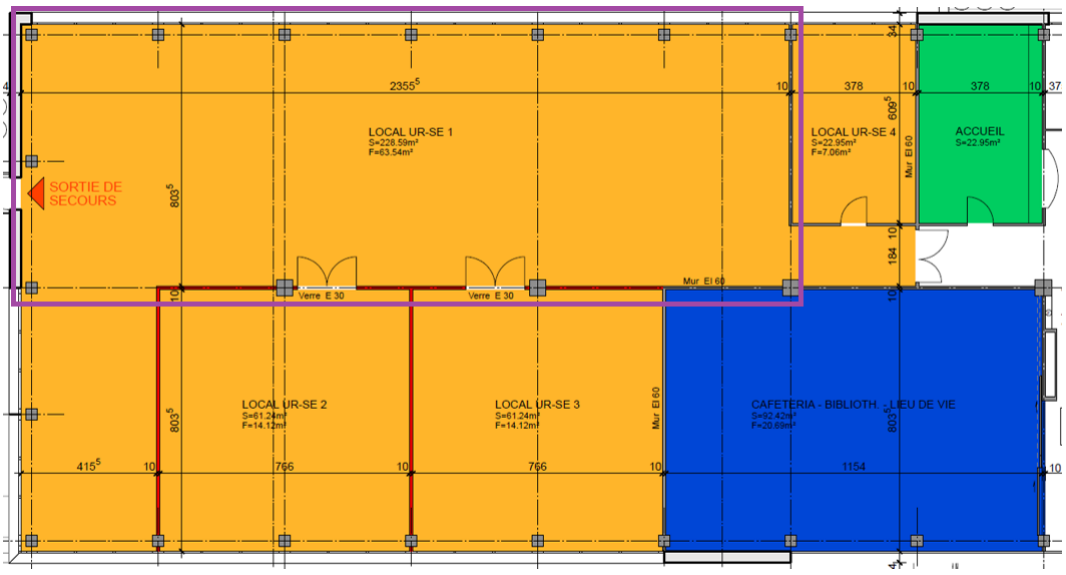


Figure 3.2 – Montre le plan réel du laboratoire

3.2 Programme de prise de mesure [Mic17]

Ce chapitre documente le programme de prise de mesure qui avait été développé. Ce programme permet avant tout d'exploiter pleinement le protocole de localisation développé, ainsi que les algorithmes de résolution de position.

Le système peut être composé de un ou plusieurs esclaves et de un ou plusieurs maîtres. Les espions font bien entendu partie intégrante également. Le système de localisation qui avait été développé n'utilise pas uniquement les mesures des espions, mais également celles du maître. Le programme effectue des mesures de distance, résous des positions des différents espions et les affiche dans une interface graphique. Le tout est configurable au travers d'un simple fichier de configuration au format texte.

Dans le cadre de ce projet, le fichier de configuration (stage.cfg) contient les informations (position, id,...) du "maitre" ainsi que des quatres esclaves.

Le programme réalisé pour ce démonstrateur a du être adapté car il a été conçu pour effectuer des mesures en continu afin d'améliorer au fil du temps la position. Ce qui n'est pas utilisable dans le cas d'un apprentissage intelligent.

3.2.1 Architecture

Afin de mieux comprendre la structure du programme de prise de mesures, la Figure 3.3 permet de voir les blocs principaux qui interagissent dans la prise de position.

Ci-dessous, une description de chaque bloc est présentée :

1. LR24Resolver : Partie centrale du démonstrateur (chef d'orchestre). Il instancie les autres blocs et gère les différents bursts de mesure et de communication).
2. Loader : Chargé de lire le fichier de configuration et de créer un jeu d'objets utilisables facilement par les autres blocs.
3. GUI : Interface graphique du système. Il affiche la position des maîtres (carrés verts), des esclaves (cercles verts), des espions (cercles jaunes) et les distances maîtreesclave (cercles rouges).
4. PostionSolver : Chargé de résoudre la position des espions en se basant sur leurs mesures. Il utilise la méthode d'approximation aux moindres carrés avancés.

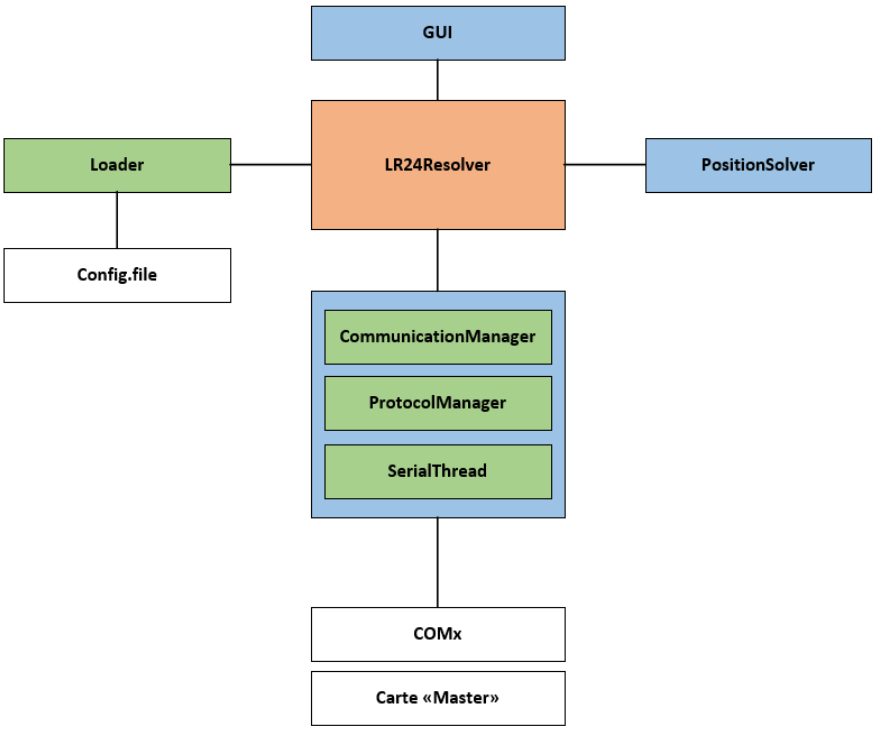


Figure 3.3 – Blocs principaux du logiciels de prise de mesure

5. SerialThread : Communication avec la carte embarquée de type maître. Ce bloc utilise le port série standard. Il lit les données disponibles et les remontes à la couche supérieure. Il transmet aussi les bytes reçus de la couche supérieur vers le port série.
6. ProtocolManager : Gestion du format du protocole de communication. Cela signifie qu'il transforme les trames reçues en jeu d'objets utilisables, ou alors, il transforme des objets reçus en une trame valide.
7. CommunicationManager : Gestion des actions (mesure, communication, calibration, etc...). Il reçois des actions et les exécute au travers du "ProtocolManager".

Un interface graphique est disponible avec le programme lors du lancement. Il est possible de démarrer une mesure, de l'interrompre et de la reseter Figure 3.4.

Ce programme est fait pour ordonner à la carte embarquée "master" d'effectuer une série de mesure. Ensuite, dès que le "master" reçoit une mesure du "spy2, il la transfère au programme qui tourne sur le PC. Les mesures réceptionnées sont sous forme de burst. Les informations utiles de ces bursts sont :

1. La provenance du burst de mesure (mesure effectuée sur quel "slave" ?)
2. La mesure brute de différence de distance pour chaque "slave" et pour 40 cannaux de fréquence différents.
3. La mesure brut du RSSI

La position du "spy" est caluclé au niveau du programme PC (PositionSolver) en fonctionne des données reçues vu précédement. Selon le fichier de configuration, les mesures sont faites sur plusieurs canaux (ranging slot number = 40). Le système va effectuer les étapes suivantes :

1. Mesurer sur 40 canaux sur un escalve
2. Récuperer les données sur les master
3. Transmettre les données au niveau du PC
4. Mesurer le deuxième esclave
5. Récupérer les données sur le master

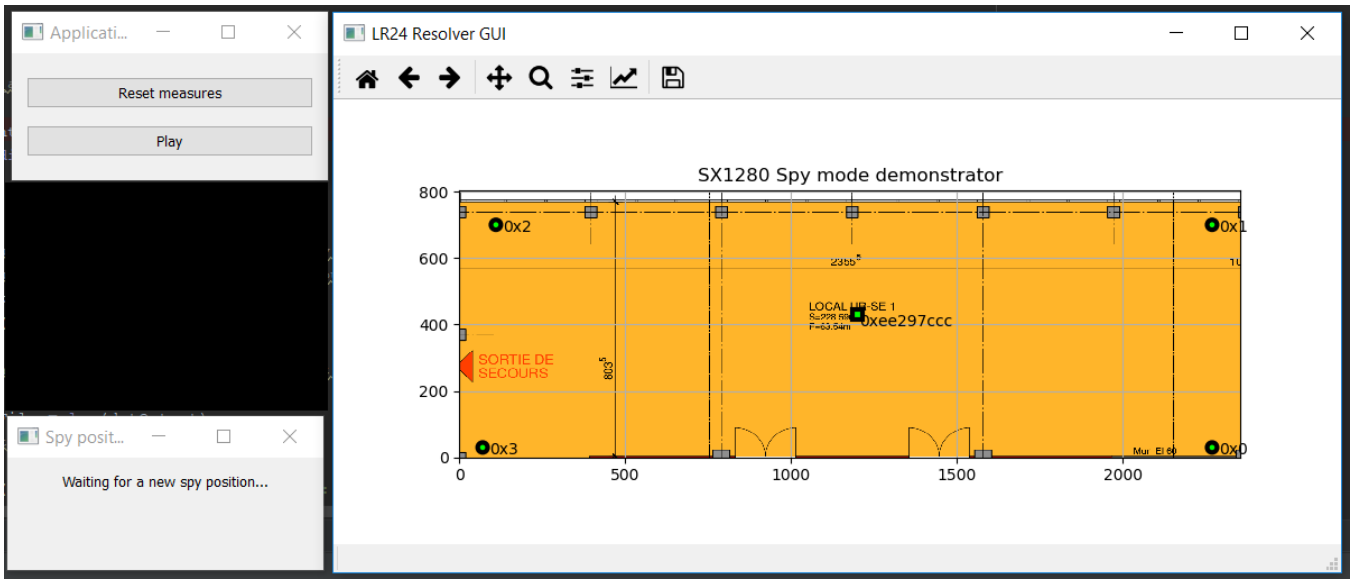


Figure 3.4 – Interface graphique du soft de prise de mesures

6. etc...
7. Estimation de position quand cela est possible

La fonction qui réceptionne les burst entrant et par conséquent la structure des données se trouvent dans le callback ci-dessous :

```
def new burst available(burst_list) :
'''
Callback when new bursts are available
:param burst_list: list_of_bursts
:return: None

=> It will calculate nodes new positions
'''
for burst in burst list:
position solver.update position_solver(burst)
system_gui.update_burst_info(burst)
```

Ce callback est reçu dans lr24_resolver.py. C'est une liste de burst (en python). En résumé, les burst reçu stock toutes les mesures pour un couple master-escalve :

valeurs[0, canal] = mesure brute pour un canal donné
valeurs[1, canal] = rssi pour un canal donné
valeurs[2, canal] = erreur pour un canal

C'est ces données qu'il est nécessaire d'exploiter pour la suite du travail.

3.2.2 Prise de données brutes

La prise de données brutes consiste à mémoriser les valeurs brutes des mesures de différences de distance fournie par l'espion. Cette mesures provient du fonctionnement des cartes embarquées qui sont configurées pour fonctionner en monde espion. Cette donnée se calcule en faisant Mspy = D1+D2-D3, voir Figure 3.5. Lorsque le système possède quatre esclave, il y aura une mesure par esclave sur chaque fois les 40 canaux. C'est avec ces dernières qu'une position est estimée.

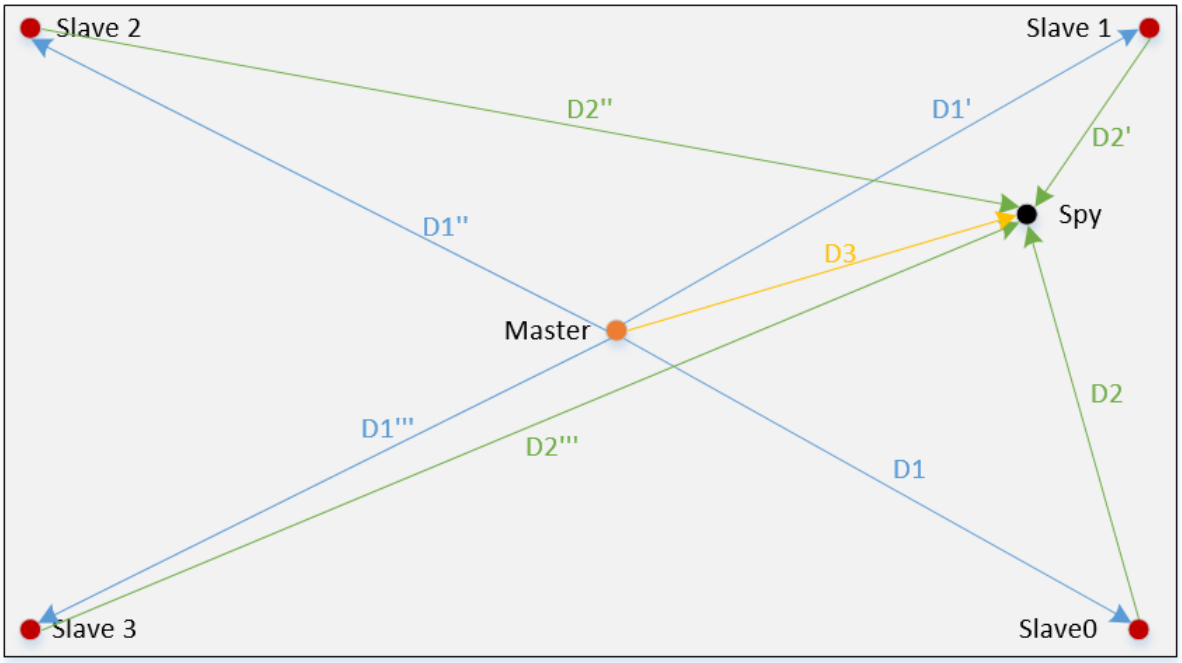


Figure 3.5 – Fonctionnement du mode spy et signification des mesures brutes

Modification du programme lr24_resolver.py

Le programme Lr24_resolver.py n'a pas pu être utilisé comme il avait été créé car il était destiné à effectuer des mesures continues afin d'améliorer au fil du temps la précision. Dans le cadre de ce projet, il est nécessaire de pouvoir maitriser la prise de mesures ainsi que gérer le nombre d'échantillons enregistrés. La Figure 3.10 montre de quel manière sont prise les données.

Il a été nécessaire de modifier la fonction new_burst_available(burst_list) du fichier Lr24_resolver.py afin de mémoriser les huit premiers burst qui arrivent depuis la carte "master". De ces "burst" sera uniquement mémorisé la valeur de la différence de distance calculée par le "spy". Cette différence de distance est obtenue grâce au mode "ranging" de la communication LoRa. Un "burst" comprend 40 informations qui sont liés aux 40 canaux de mesures.

Une fois que les 8 burst ont été reçus, le calcul de position si il existe est mémorisé puis reseté pour la mesure suivante. Il est possible de sélectionner le nombre de mesures désirés grâce à la variable measures_count_des. Quand ce nombre est atteint, le programme s'arrête automatiquement sinon il continue d'effectuer les mesures.

La fonction new_node_position_available(node_pos) a également été adaptée. Lorsqu'une mesure de position existe, il y a un contrôle qui est effectué afin de s'assurer que les mesures ont été effectué dans un odre précis qui est le suivant : Burst reçu de la mesure sur le slaveo (2x) puis du slave1 (2x) puis du slave2 (2x) et finalement du slave3 (2x). Cela afin de ne pas mélanger les données lors de l'apprentissage à l'aide de l'algorithme SVM.

Finalement, pour que les mesures soient automatique avec le moins d'interaction possible avec l'utilisateur le fichier gui.py a été modifié pour y ajouter les fonctions software des boutons "stop", "start" et "reset".

Structure des données

Dans la Figure 3.7 est représenté comment sont structurées les mesures récupérées par le programme python (Lr24_resolver.py). Il a été nécessaire de faire une réflexion concernant quoi prendre comme mesure

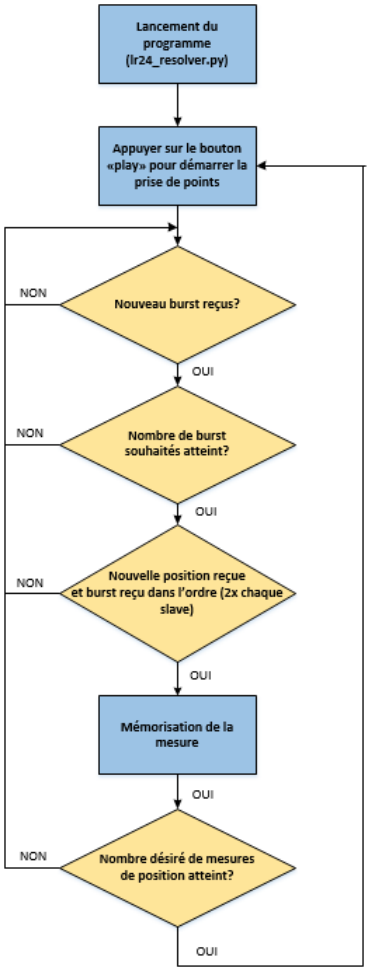


Figure 3.6 – Diagramme expliquant la prise de mesures

et de quelle manière. Il a été décidé de récupérer les informations ci-dessous. Un fichier est créé par endroit de mesure.

1. Index : Numéro de la mesure pour un point donné
2. X1 : Coordonnée X réelle par rapport aux repères de la salle
3. Y1 : Coordonnée Y réelle par rapport aux repères de la salle
4. X2 : Coordonnée X calculée par le programme python Lr24_resolver.py après avoir reçus 8 bursts
5. Y2 : Coordonnée Y calculée par le programme python Lr24_resolver.py après avoir reçus 8 bursts
6. meas : Données brutes de la valeur de "ranging" sur les 40 canaux reçues à chaque burst (8x40 valeurs)

Une erreur a été commise à ce niveau. Il aurait été très utiles de mémoriser également la valeur du RSSI. Chose qui n'a pas été faite à ce stade de la recherche. Effectivement, la prise de mesures est plus complexe qu'espérée et prend passablement de temps. Il avait d'abord été décidé de faire le test sans le RSSI et de l'intégrer par la suite en pensant qu'une mesures était rapidement faite ce qui n'a pas été le cas. Comme le travail étant de courte durée, il a été tout de même nécessaire d'avoir des données pour effectuer le travail et j'ai du me concentrer sur les données disponible sans la valeur du RSSI.

Sur la Figure 3.7 la partie "meas" est séparée en quatre parties car les bursts sont reçus dans l'ordre deux fois pour l'esclave 0, deux fois pour l'esclave 1, deux fois pour l'escalve 2 et finalement deux fois pour l'esclave 3. C'est afin de mieux comprendre malgré que les mesures se trouvent les unes derrières les autres.

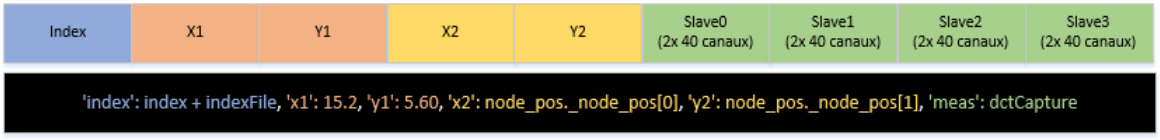


Figure 3.7 – Structure des données

3.2.3 Prise de données de positions convergées

La prise de données des positions convergées est similaire à la prise des mesures brutes sauf que lors du lancement de la mesure, il faut laisser la mesure converger sur sa position finale et la mémoriser sans tenir compte des mesures brutes des différents canaux.

Cette prise de mesure supplémentaire a été réalisée car les résultats obtenus uniquement avec les valeurs brutes ne sont pas satisfaisants (voir dans le chapitre suivant).

Modification du programme lr24_resolver.py

Pareil que ci-dessus, le programme Lr24_resolver.py n'a pas pu être utilisé comme il avait été créé car il était destiné à effectuer des mesures continues afin d'améliorer au fil du temps la précision. Dans le cadre de ce projet, il est nécessaire de pouvoir maitriser la prise de mesures. La Figure 3.8 montre de quel manière sont prise les données.

Le modifications apportée au logiciel sont similaires à celles détaillées ci-dessus. La seule différence c'est qu'il a été nécessaire de regarder à partir de quel moment la mesure converge et effectuer ce temps de mesures pour chaque point. La mesure dure environ 3min pour obtenir une position convergée ce qui correspond à la réception de 120 burst.

Le résultat de la position est mémorisé à la suite des 120 burst.

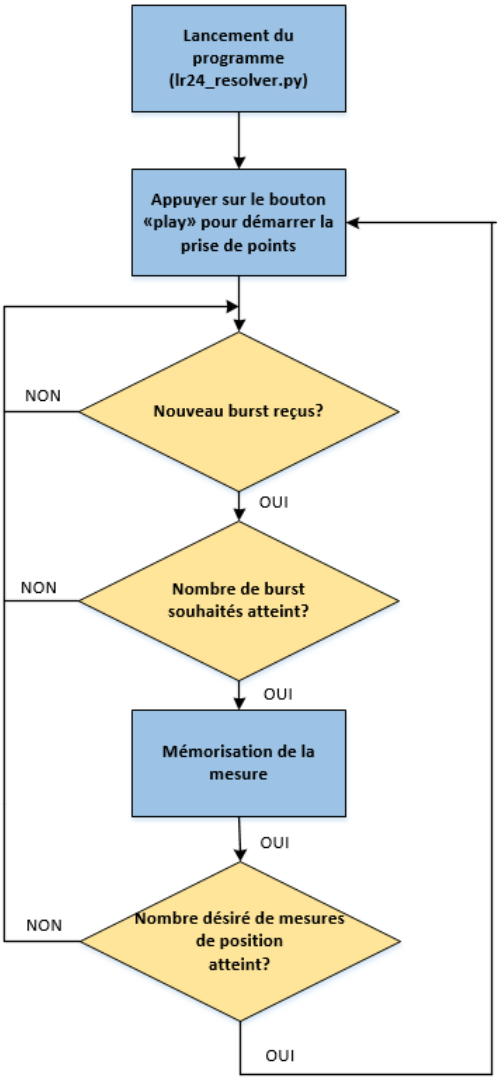


Figure 3.8 – Diagramme expliquant la prise de mesures

Structure des données

La structure des données voir la Figure 3.9 est similaire à la partie ci-dessous saut que les données brutes ne sont pas mémorisée car elles seraient trop importantes.

1. Index : Numéro de la mesure pour un point donné
2. X1 : Coordonnée X réelle par rapport aux repères de la salle
3. Y1 : Coordonnée Y réelle par rapport aux repères de la salle
4. X2 : Coordonnée X calculée par le programme python Lr24_resolver.py après avoir convergée (120 bursts)
5. Y2 : Coordonnée Y calculée par le programme python Lr24_resolver.py après avoir convergée (120 bursts)

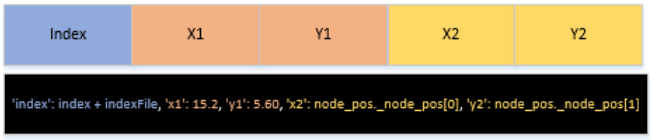


Figure 3.9 – Structure des données contenant uniquement la valeur x/y calculée

3.3 Plan de mesure

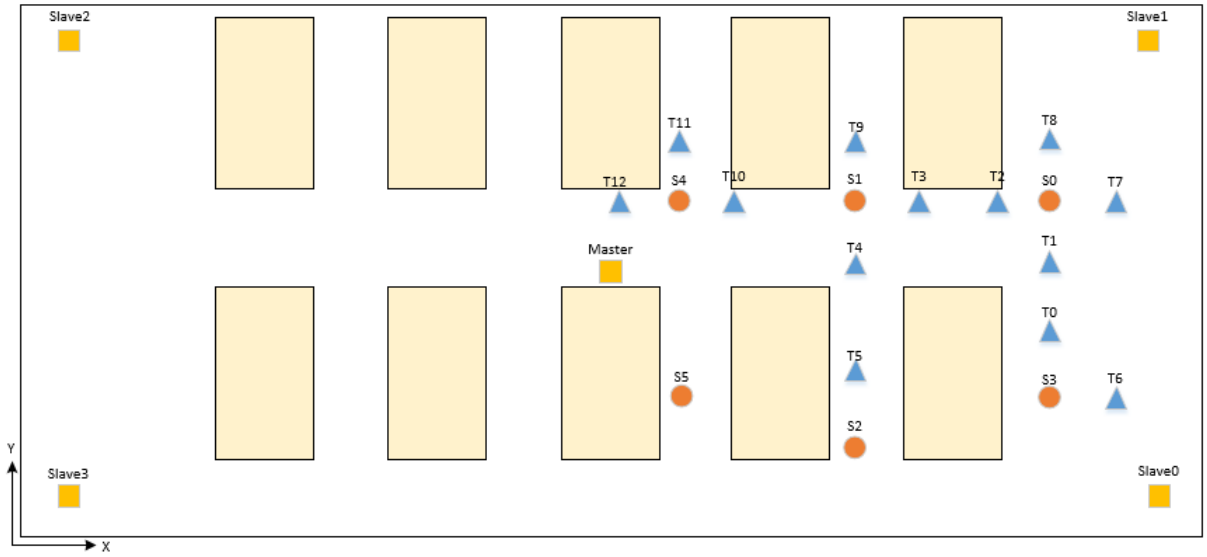
Un plan de mesure idéal avait été imaginé dans un premier temps pour effectuer les premiers essais, voir sur la Figure 3.1. Ce plan n'a pas pu être réalisé exactement de cette manière dû à la configuration de la salle et de la disposition des établis. Comme décrit ci-dessus, la configuration de la salle pour effectuer les mesures est composée de quatre esclaves (slaves) placés dans les coins de la pièce à mesurer et le maitre (master) est placé au centre. Ensuite, l'espion est déplacé à différents endroit pour effectuer la prise de mesures voir sur la Figure 3.10.

Afin de mieux comprendre la méthodologie de mesures, un cas réel a été imaginé. C'est-à-dire, que la réflexion a été faite en partant du principe qu'il faut positionner des objets dans un laboratoire et installer le système sans effectuer un quadrillage trop précis afin que le temps d'action soit le plus court possible.

Les points marqués par Sx (orange foncé) sont les points sur lesquels le système est basé et correspondent à la position de l'espion (spy). Quarante mesures ont été effectuées sur ces six endroits et c'est eux qui seront utilisé pour entrainer le système. Cinq mesures supplémentaires ont été effectuées à ces endroits pour faire des vérifications. Ces mesures ont été effectuées à des moments différents à l'exception de S5.

Les triangles bleus marqués Tx correspondent à des points de tests et cinq mesures ont été effectuées par point.

A noter que toutes ces mesures on été effectuée en positionnant un mat muni d'un carton pour poser la carte de l'espion. Donc à chaque changement de position, il est possible que cette dernière varie de quelques centimètres et que l'orientation de l'antenne soit différente.



Taille de de la salle : x = 23.55m / y = 8.03m

Slave0 : x =22.70m / y = 0.30m
Slave1 : x = 22.70m / y = 7.00m
Slave2 : x = 1.10m / y = 7.00m
Slave3 : x = 0.70m / y = 0.30m
Master : x = 12.00m / y = 4.30m

S0 : x = 21.30m / y = 5.60m
S1 : x = 17.30m / y = 5.60m
S2 : x = 17.30m / y = 2.10m
S3 : x = 21.30m / y = 3.10m
S4 : x = 13.80m / y = 5.60m
S5 : x = 13.80m / y = 3.10m

T0 : x = 21.30m / y = 4.00m
T1 : x = 21.30m / y = 5.00m
T2 : x = 20.50m / y = 5.60m
T3 : x = 18.50m / y = 5.60m
T4 : x = 17.30m / y = 5.00m
T5 : x = 17.30m / y = 3.50m
T6 : x = 22.00m / y = 3.10m
T7 : x = 22.00m / y = 5.60m
T8 : x = 21.30m / y = 6.60m
T9 : x = 17.30m / y = 6.60m
T10: x = 14.80m / y = 5.60m
T11: x = 13m80 / y = 6.60m
T12: x = 12.80m / y = 5.60m

Figure 3.10 – Plan des mesures effectuées et du setup

4 Traitement des données et classification

Comme décidé lors de l'analyse de l'état de l'art et les réflexions de départ, la premières analyse qui est faite est de faire de la classification avant de tester les résultats de la régression.

L'algorithme qui a été sélectionné pour effectué cette tâche est SVM (Support Vector Machines). Ce chapitre va décrire les différentes étapes qui ont été réalisée pour traiter les données et utiliser l'algorithme.

La programmation a été réalisée sur l'environnement PyCharm PROFESSIONAL 2019.2 et dans le langage Python. Cet environnement est le plus familier pour moi car il est étudié dans différents cours c'est pourquoi mon choix c'est porté sur ce dernier.

4.1 Structure du traitement

Pour le traitement de données pour un apprentissage à l'aide d'un algorithme il est conseillé d'avoir différentes étapes voir Figure 4.1.

1. Acquisition des données : Étape délicate et très importante dans ce projet.
2. Pré-traitement : Étape de traitement sur les données brutes se trouvant dans le set de données.
3. Extraction des caractéristiques : Étape qui consiste à extraire les caractéristiques qui seront utilisée par l'algorithme pour faire la reconnaissance.
4. Reconnaissance : Utilisation d'un algorithme pour effectuer la reconnaissance des positions.
5. Décision : Cette étape consiste, si cela est nécessaire, à décider du résultat final à l'aide d'une fusion, cela n'a pas été utilisé pour l'instant.

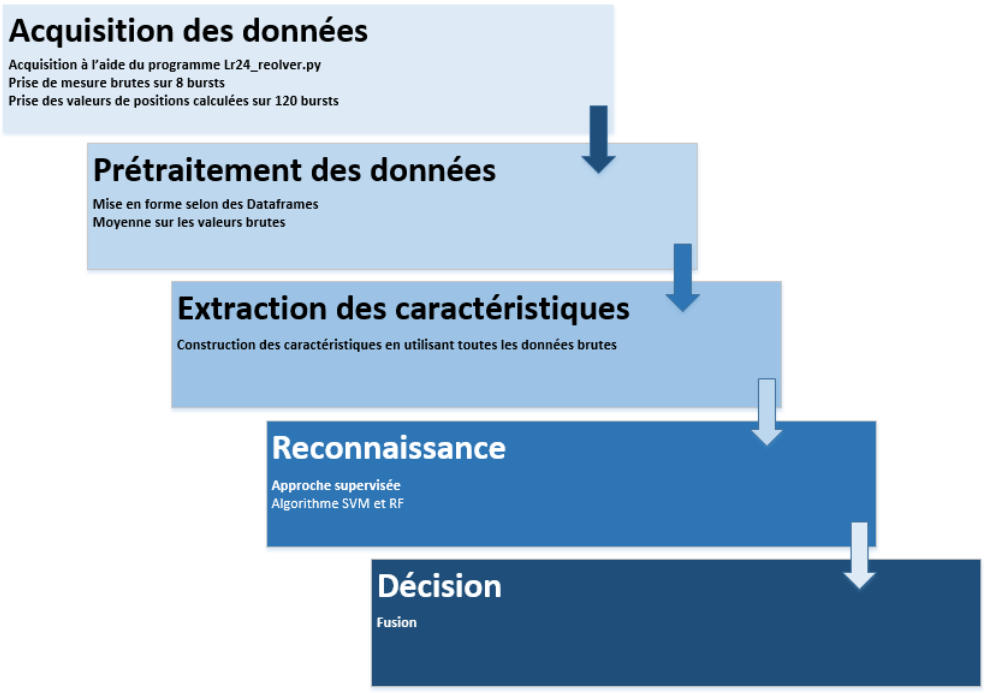


Figure 4.1 – Flux de traitement de l'information

Le premier point et l'acquisition de données qui est décrit au chapitre précédent. C'est une étape primordiale car c'est grâce à ces données que toute la suite de l'analyse va dépendre. Dans le cadre de ce projet, on remarque vite que si les données ne sont pas prises de manière rigoureuses, cela ne fonctionne pas.

Le pré-traitement consiste à utiliser les données disponible et de les exploiter. Dans un premier temps les données sont lue depuis un fichier *.npy qui représente des données NumPy. NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes [Wikc]. Depuis, ce fichier plusieurs traitement son effectué afin de transformer les données et les mettre dans un format permettant un traitement simple, le format "Dataframe" a été choisi et appartient à la librairie Pandas. Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles [Wikd]. A partir de là, il est possible d'utiliser les données afin d'en extraire des caractéristiques qui est l'étape suivante. Ce point sera détaillé plus loin afin de mieux comprendre ce qui a été utilisé pour que l'algorithme puisse au mieux reconnaitre les différentes positions.

L'étape de reconnaissance consiste à entrainer un système avec les caractéristiques sélectionnées à partir des données et ensuite de valider que notre algorithme est capable de reconnaitre de nouvelles mesures. Ce point est également détaillé plus bas.

Finalement, dans ce type d'analyse, il existe souvent une partie décision. Cela peut se faire car deux algorithme différents sont utilisées et il est nécessaire de décider quelle est le meilleur résultat. Dans le cadre de ce projet il n'a pas été nécessaire d'utiliser cela car il n'a pas été possible dans le temps imparti de tester plusieurs technique et donc la fusion n'était pas utile.

4.2 Extraction des caractéristiques

Ce chapitre traite des caractéristiques qui seront utilisées par l'algorithme afin de déterminer à quelle classe elles appartiennent. La Figue 4.2 montre comment se présentent les données. Il est possible de les séparer et d'isoler chaque mesure. Chaque mesures est composée des données pour les quarante canaux (colonne : Canal).Les mesures qui peuvent être exploitées pour effectuer la classification est les mesures brutes de différence de distance (Sx.x). Il serait aussi possible d'exploiter la position fournie par le programme de prise de mesure mais cette dernière n'est pas assez représentative. C'est pour cette raison que deux type de classification ont été effectués.

La première consiste à utiliser les données brutes et de travailler avec ces dernières pour extraire d'autres caractéristiques. Les principales qui ont été testée sont décrites ci-dessous.

1. Toutes les données (raw) : Cela consiste à prendre toutes les mesures brutes de tous les canaux et de les mettre les unes après les autres. Ce qui veut dire que ca fait 320 valeurs (40 canaux fois 2x chaque esclave).
2. Un canal (raw) : Cette façon de faire est identique à la précédente saut qu'un seul canal est pris en compte ce qui fait 8 valeurs (1 canal fois 2x chaque esclave).
3. Moyenne : La moyenne est faite sur les 40 canaux par slave ce qui va donner huit caractéristiques supplémentaires.
4. Variance : La variance est une mesure qui permet de caractériser la dispersion d'un échantillon. Elle est utilisée pour voir la dispersion des valeurs par esclave sur les quarante canaux.
5. Déviation standard : La déviation standard équivaut à la racine carrée de la variance. C'est donc une mesure de dispersion qui est faite également par esclave sur les quarante canaux.
6. Quantile : Un quantiles correspond à séparer les données en partie de taille égale. C'est-à-dire que chaque partie doit contenir le même nombre de données. Cela est également utilisé pour séparer les données au niveau des canaux pour un esclave.
7. Médiane : La médiane est en quelque sorte le quantile qui sépare des données en deux partie de même taille.

8. covariance : La covariance permet de voir la corrélation entre des variables. Cette mesure est effectuée par esclave sur tous les canaux.

La deuxième consiste à utiliser uniquement la position fournie par le programme de prise de mesure. Cette position est mémorisée après avoir convergé et ce qui fait que les caractéristiques disponibles sont la coordonnée X et la coordonnée Y. Comme il n'y a que deux caractéristiques, il sera difficile d'effectuer différent traitement sur ces données si l'algorithme n'arrive pas différencier les classes.

Les résultats de ces deux différentes approches seront détaillées au chapitre suivant.

	Pos	Meas	Canal	s0.0	s0.1	s1.0	s1.1
25	0.00000	0.00000	25.00000	147.00000	-1136.00000	1334.00000	-739.00000
26	0.00000	0.00000	26.00000	701.00000	-26.00000	838.00000	394.00000
27	0.00000	0.00000	27.00000	615.00000	-77.00000	460.00000	-324.00000
28	0.00000	0.00000	28.00000	510.00000	298.00000	715.00000	255.00000
29	0.00000	0.00000	29.00000	497.00000	-406.00000	1277.00000	-259.00000
30	0.00000	0.00000	30.00000	598.00000	-137.00000	614.00000	514.00000
31	0.00000	0.00000	31.00000	297.00000	-20.00000	465.00000	-28.00000
32	0.00000	0.00000	32.00000	780.00000	1869.00000	-358.00000	-388.00000
33	0.00000	0.00000	33.00000	901.00000	162.00000	384.00000	-511.00000
34	0.00000	0.00000	34.00000	1110.00000	260.00000	763.00000	5.00000
35	0.00000	0.00000	35.00000	1067.00000	498.00000	-163.00000	-492.00000
36	0.00000	0.00000	36.00000	1189.00000	383.00000	718.00000	-230.00000
37	0.00000	0.00000	37.00000	1.00000	450.00000	328.00000	-107.00000
38	0.00000	0.00000	38.00000	1117.00000	-14.00000	823.00000	173.00000
39	0.00000	0.00000	39.00000	-144.00000	61.00000	1721.00000	329.00000
40	0.00000	1.00000	0.00000	1274.00000	848.00000	829.00000	373.00000
41	0.00000	1.00000	1.00000	993.00000	299.00000	1221.00000	-9.00000
42	0.00000	1.00000	2.00000	993.00000	-847.00000	1235.00000	-292.00000
43	0.00000	1.00000	3.00000	986.00000	581.00000	801.00000	47.00000
44	0.00000	1.00000	4.00000	424.00000	-143.00000	1652.00000	208.00000
45	0.00000	1.00000	5.00000	226.00000	394.00000	1959.00000	1062.00000
46	0.00000	1.00000	6.00000	794.00000	1384.00000	1785.00000	1646.00000
47	0.00000	1.00000	7.00000	989.00000	480.00000	1630.00000	1124.00000
48	0.00000	1.00000	8.00000	925.00000	-165.00000	2212.00000	1327.00000
49	0.00000	1.00000	9.00000	1044.00000	528.00000	877.00000	220.00000

s2.0	s2.1	s3.0	s3.1	x1	y1	x2	y2
826.00000	444.00000	1175.00000	-1379.00000	21.30000	5.60000	21.01397	6.65830
400.00000	469.00000	1169.00000	0.00000	21.30000	5.60000	21.01397	6.65830
1932.00000	776.00000	2558.00000	1646.00000	21.30000	5.60000	21.01397	6.65830
1373.00000	1175.00000	1192.00000	698.00000	21.30000	5.60000	21.01397	6.65830
485.00000	338.00000	1323.00000	480.00000	21.30000	5.60000	21.01397	6.65830
446.00000	1422.00000	1025.00000	1260.00000	21.30000	5.60000	21.01397	6.65830
1051.00000	693.00000	330.00000	-182.00000	21.30000	5.60000	21.01397	6.65830
988.00000	1181.00000	817.00000	1416.00000	21.30000	5.60000	21.01397	6.65830
438.00000	390.00000	778.00000	463.00000	21.30000	5.60000	21.01397	6.65830
867.00000	540.00000	485.00000	537.00000	21.30000	5.60000	21.01397	6.65830
667.00000	1584.00000	490.00000	988.00000	21.30000	5.60000	21.01397	6.65830
-383.00000	283.00000	1022.00000	572.00000	21.30000	5.60000	21.01397	6.65830
727.00000	884.00000	760.00000	1232.00000	21.30000	5.60000	21.01397	6.65830
497.00000	842.00000	891.00000	983.00000	21.30000	5.60000	21.01397	6.65830
909.00000	1116.00000	1006.00000	1154.00000	21.30000	5.60000	21.01397	6.65830
182.00000	798.00000	-26.00000	1962.00000	21.30000	5.60000	21.16969	3.75498
860.00000	285.00000	670.00000	1013.00000	21.30000	5.60000	21.16969	3.75498
974.00000	907.00000	703.00000	965.00000	21.30000	5.60000	21.16969	3.75498
1108.00000	791.00000	1139.00000	726.00000	21.30000	5.60000	21.16969	3.75498
1015.00000	746.00000	531.00000	633.00000	21.30000	5.60000	21.16969	3.75498
941.00000	688.00000	408.00000	1465.00000	21.30000	5.60000	21.16969	3.75498
1386.00000	1077.00000	1082.00000	768.00000	21.30000	5.60000	21.16969	3.75498
633.00000	847.00000	348.00000	990.00000	21.30000	5.60000	21.16969	3.75498
643.00000	-2748.00000	181.00000	482.00000	21.30000	5.60000	21.16969	3.75498
1466.00000	1270.00000	899.00000	1050.00000	21.30000	5.60000	21.16969	3.75498

Figure 4.2 – Données sous forme de dataframe pour le traitement et l'extraction des caractéristiques

4.3 Reconnaissance - Algorithme

Le choix de l'algorithme utilisé a été fait par rapport à l'état de l'art qui a été effectué en début de projet. Comme déjà mentionné, l'algorithme qui a été retenu et SVM (Support Vector Machine). Cette solution permet de laisser le projet évoluer en utilisant dans un premier temps la solution pour la classification et ensuite en utilisant SVM pour la régression.

Support vector machine est un algorithme supervisé qui permet de faire de la classification en essayant de trouver le meilleur hyperplan. La librairie scikit learn a été utilisée et je me suis servie de sklearn.svm.SVC (System Vector Classification).

L'implémentation de l'algorithme est facile une fois que les labels et les caractéristiques ont été extraits. Dans un premier temps, le canevas de base a été mis en place en utilisant les données brutes des mesures de "ranging". Cela a permis de faire les premiers réglages.

Afin d'entrainer et de tester l'efficacité de l'algorithme il est nécessaire de partager les données en trois set. Il y a un set qui est utilisé pour l'entrainement, un set qui est utilisé pour valider l'entrainement qui vient d'être réalisé et un troisième set qui sert uniquement pour le résultat final. Dans ce dernier set, se sont des données que l'algorithme n'a jamais vu. La Figure 4.3 permet d'illustrer cette séparation. IL est également possible de voir que les données d'entrainement et de validation vont de pair. Pour ma part j'ai décidé d'effectuer une cross-validation. C'est-à-dire qu'à chaque lancement le partage des données n'est jamais identique. La seule chose qui est respectée est qu'il y aura toujours 20% de données pour la validation et 80% pour l'entrainement. Un autre chose qui est a vérifier c'est que les dataset soit balancé. Cela veut dire qu'il doit y avoir le même nombre de données de chaque classe dans chaque set. Cela donnerait de mauvais résultats si l'entrainement est fait avec les classes 1,2,et 3 et que la validation se fait avec des données des classes 4 et 5 par exemple.

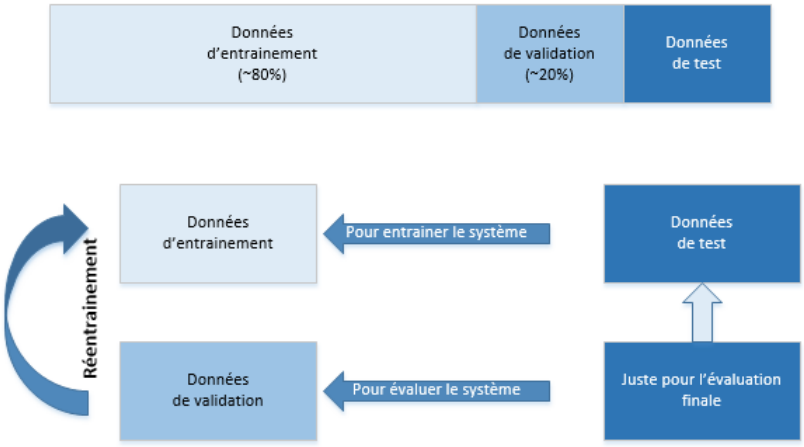


Figure 4.3 – Partage des données

4.3.1 hyper-paramètres

Il a été nécessaire de chercher les meilleurs paramètres pour SVM. Seul le kernel, le paramètre C et le paramètre gamma ont été cherchés. Une forte valeur de C tente de minimiser les erreurs de classification des données d'entraînement et une faible valeur essaie de maintenir une classification lisse. Concernant le gamma, plus il est grand , plus la cloche de la gaussienne est étroite voir Figure 4.4.

La première chose qui a été faite a été de trouver le meilleur kernel pour cette classification. Dans mon cas, j'ai essayé les kernel suivant : linear, rbf, poly, sigmoid. Celui qui a donnée les meilleurs résultats et le kernel linear.

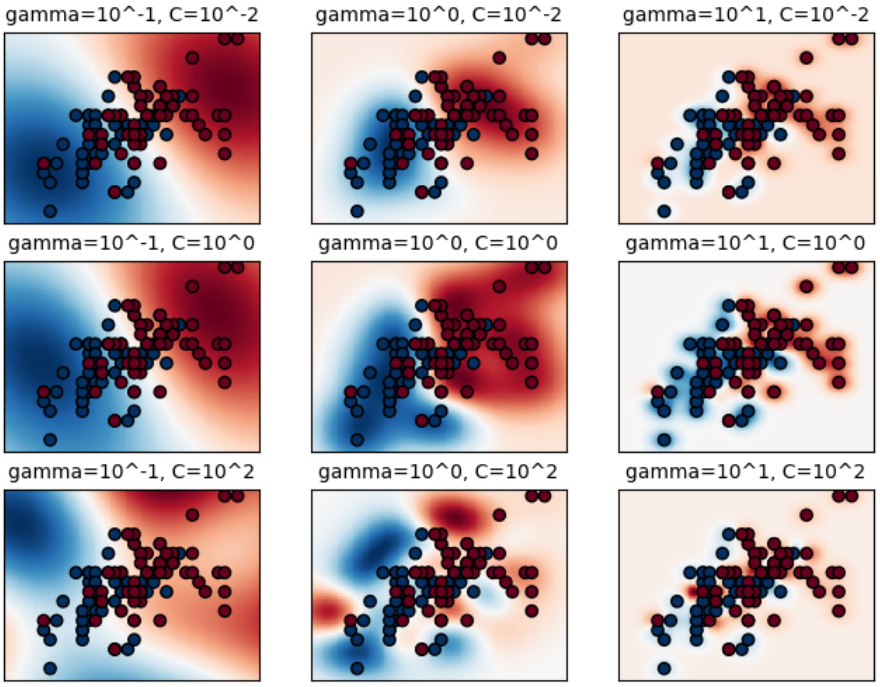


Figure 4.4 – Influence des paramètres C et gamma

Suite à cela, un grid-search est effectué sur les paramètres C et gamma. Les meilleurs résultats obtenus pour la feature de la durée de charge sont avec un C égal à 100 et un gamma égal à 0.001.

4.4 Validation et résultats

Cette section va présenter l'évolution des résultats obtenus au cours du développement. Voici un résumé des données acquises selon la Figure 3.10 :

1. Mesures de "ranging" effectuées sur les points Sx qui correspondent aux points que l'on souhaite retrouver (40x sur chaque point).
2. Mesure de "ranging" effectuées sur les points Sx à un moment différent que les 40 mesures précédentes (5x sur chaque point).
3. Mesure de "ranging" effectuées sur les points Tx. Ces points sont pris afin de voir si le plus proche voisin est retrouvé.
4. Mesure de position effectuées sur les points Sx (40x + 5x sur chaque point).
5. Mesure de position effectuées sur les points Tx (5x sur chaque point)

Ces différentes prises de données permettent de faire plusieurs analyses et comprendre de façon plus clair comment le système se comporte.

Les résultats sont évalués à l'aide des matrices de confusion mais également à l'aide de la mesure de précision, du F1-Score macro et finalement du F1-Score micro.

La Figure 4.5 montre ce que représente la précisoin et le rappel. Ces deux notions sont utilisées pour calucler le F1-Score de la manière suivante :

F1_score = 2((Précision * rappel)/(Précision + rappel))

Le micro ou macro-avarage est utilisé pour analyser les données de manière différente. Pour le calcul du macro-avarage, il faut aditionner les précisions de chaque classe et le diviser par le nombre de classe. Dans ce cas il se pourrait qu'une classe soit très mauvaise et déséquilibrée mais que les autres remonte le score. Si

maintenant on parle du micro-avarage, le calcule se fait en additionnat les vrais positif et en les divisant par le nombre total de point. Cette manière de faire va donner un score plus bas si une classe est déséquilibrée [Dat].

Dans le cadre de cette analyse, toutes les classes possèdent le même nombre de données.

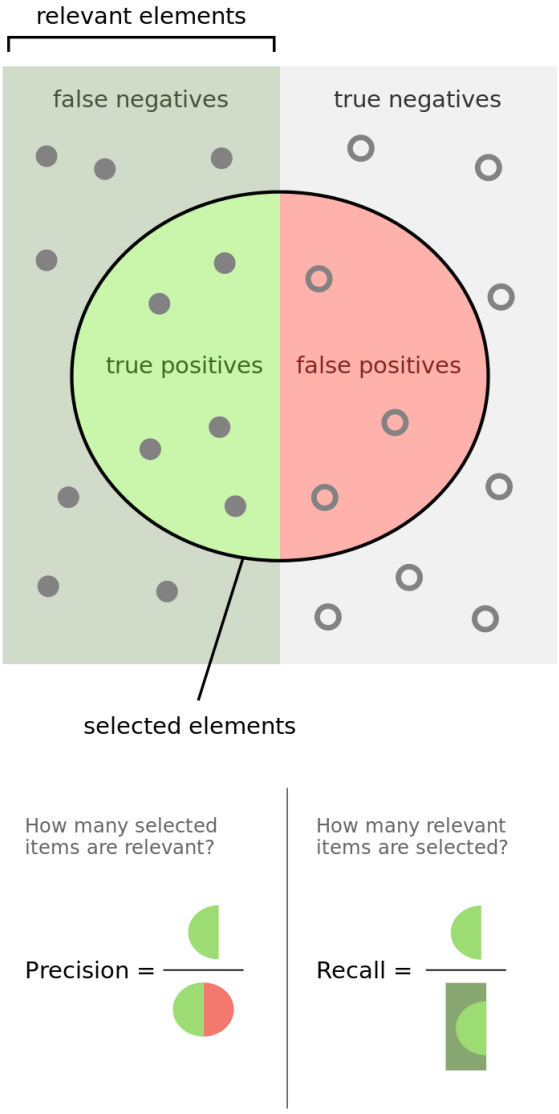


Figure 4.5 – Repérsentation de la précision et du rappel [**WIKI**]

4.4.1 Résultats en utilisant les données de "ranging"

Ce chapitre va présenter les résultats qui ont été obtenus à partir des données brutes de "ranging" fournies par le programme de mesures. Plusieurs traitements sont faits sur ces mesures afin d'améliorer les résultats. Le calule de position est également disponible mais n'est pas très représentatif car calculé uniquement à partir de huit valeurs de ranging (deux sur chaque esclave).

Afin d'avoir une meilleure vue de ce que représente les positions calculées sur peu d'échantillon, la Figure 4.6 montre le positionnement de ces points calculés par rapport à la position réelle de l'espion. Les gros points de couleurs représentent la position réelle et les petits points de couleurs représentent les positions calculées. La croix noir représente la position du master. On remarque que les petits points sont bien

plus espacés que lorsque l'on laisse converger la position (Figrure 4.15) mais il y a tout de même une séparation.

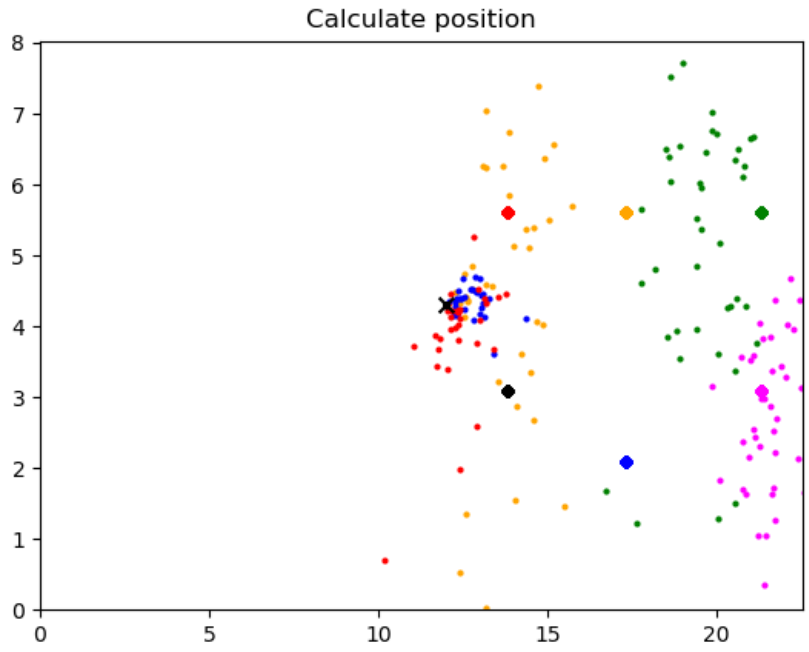


Figure 4.6 – Représentation des points non convergés par rapport aux positions réelles

La Figure 4.6 met en évidence un phénomène qui tend à faire rapprocher les points sur la position du maitre. Cet effet qui se trouve dans le laboratoire où les mesures ont été effectuées complique la reconnaissance de position.

Ci-dessous, une énumeration des couleurs et de la position associée :

1. Couleur verte (SO) : x = 21.30m / y = 5.60m
2. Couleur jaune (S1) : x = 17.30m / y = 5.60m
3. Ccouleur bleue (S2) : x = 17.30m / y = 2.10m
4. Couleur rose (S3) : x = 21.30m / y = 3.10m
5. Couleur rouge (S4) : x = 13.80m / y = 5.60m
6. Couleur noir (S5) : x = 13.80m / y = 3.10m

Comme décrit plus haut, toute une série de tests ont été effectués afin de trouver d'une part les meilleurs paramètre et d'autre part de trouver les caractéristiques les plus probantes. Dans tous les essais realisés, seul trois seront détaillés ci-dessous :cRésultats obtenus en utilisant uniquement les données de "ranging", résultats ontenus en utilisant uniquement les données de position et finalement présentation de la solution retenue qui utilise la mediane, les données "ranging", les quantiles et la variance.

Résultat en utilisant toutes les données de ranging brutes

Le premier test qui a été effectué a été de prendre toutes les données brutes des valeurs de ranging. Cela consiste à prendre chacune des mesures des quatre esclaves sur tous les quarantes cannaux et de les mettre les unes à côté des autres pour en créer une caractéristique. Cela est effectué pour chacune des mesures (40 par position).

La Figure 4.7 présente le résultat obtenu lorsque l'entrainement est fait avec 32 des 40 mesures sur une même position et testé avec les 8 restants. Le résultat obtenu est impressionnant et offre une précision entre

95% et 100%. Les position sont très bien reconnues. La variation de pourcentage pour la reconnaissance vient du fait qu'à chaque lancement, le programme exclu 8 autres mesures.

La matrice de confusion représentée est donnée avec des mesures qui ont été faites sur un même point sans bouger l'espion. Le F1-score macro est de 0.9791 et un F1-score micro de 0.9792.

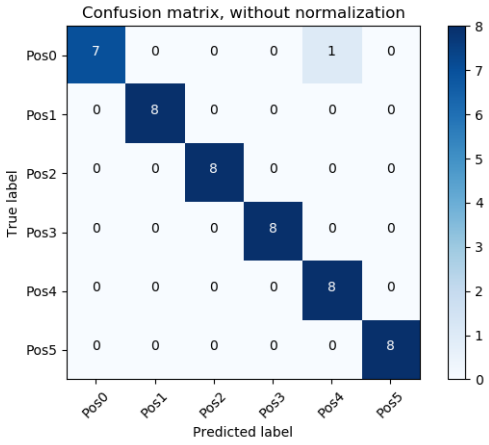


Figure 4.7 – Matrice de confusion obtenue en utilisant les point Sx et les caractéristiques de "ranging"

Afin de valider le système, 5 points suplémentaires par postion ont été pris durant une autre période et en ayant propablement une postion à peine différentes. Il faut précisier que les mesures sur la position 5 ont été effectuées au même moment tant pour les 40 que les 5 mesures. C'est pour cette raison que la détection de cette position est très bonne.

Cela montre une précision de 46.67% ce qui correspond à 14 positions correctement reconnues sur 30. Ces résultats ne sont pas encourageant car cela veut dire que sitôt que l'espion est positionné différement mais au même endroit cela influence grandement la reconnaissance. Il est donc nécessaire d'effectuer un travaille supplémentaire pour améliorer cette reconnaissance soit niveau prise de mesure soit niveau du travail sur les caractéristiques. Ce qui a été obtenu dans ce test n'était pas ce qui était attendu au vu des résultats obtenus précédement dans la Figure 4.7.

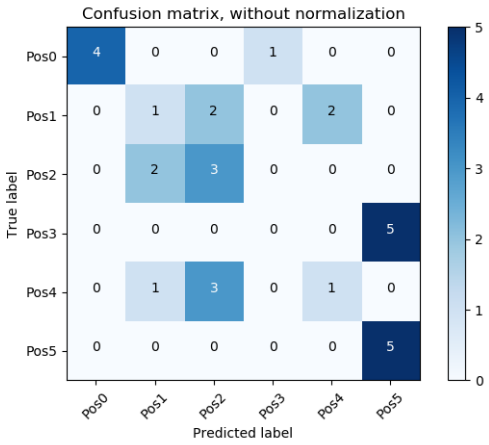


Figure 4.8 – Matrice de confusion obtenue en utilisant les point de tests SxT et les caractéristiques de "ranging" de la 1ère et 2ème mesures de l'esclave

Deux mesures sont effectuées par esclave pour les quarantes cannaux. L'exemple précédent prend en compte toutes ces mesures. Afin de voir l'effet de ces mesures, un essai a été de prendre uniquement chaque première mesure et ensuite chaque seconde mesure. Etonnement cela à amélioré les résultats comme le

montre la Figure 4.9. Les premières mesures offre de mauvais résultats alors que prendre uniquement les deuxièmes mesures améliore le score. La précision obtenue est de 53.33% ce qui correspond à 16 positions correctement reconnues sur 30. Difficile à conclure le pourquoi de cette amélioration, afin de clarifier cette influence il serait intéressant de faire plus de deux mesures et regarder les différents résultats.

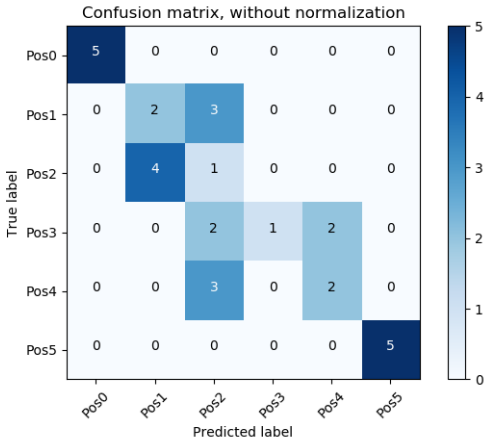


Figure 4.9 – Matrice de confusion obtenue en utilisant les point de tests SxT et les caractéristiques de "ranging" de la 2ème mesure de l'esclave uniquement

Résultat en utilisant la position non convergée

Le résulat qui est présenté ici permet de faire une comparaison entre une mesure qui est convergée (présenté au chapitre suivant) ou une mesure qui est calculé uniquement à partir de huit mesure. C'est sans surprise que les résultats qui sont obtenus sont vraiment moins bons.

La Figure 4.11 présente le résultat obtenu lorsque l'entrainement est fait avec 32 des 40 mesures sur une même position et testé avec les 8 restants. Le résultat obtenu est impressionnant et offre une précision d'environ 70% contre environ 85% pour la solution avec les positions convergées.

La matrice de confusion représentée est donnée avec des mesures qui ont été faites sur un même point sans bouger l'espion. La précision est de 72.91%, le F1-score macro est de 0.7102 et un F1-score micro de 0.7293. 83.33

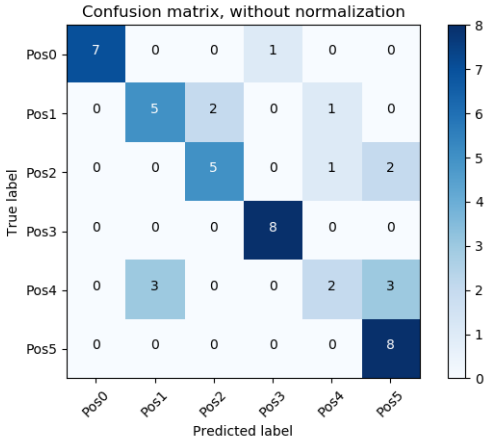


Figure 4.10 – Matrice de confusion obtenue en utilisant les point Sx et la caractéristique de position

Afin de valider le système, 5 points suplémentaires par postion ont été pris durant une autre période et en ayant propablement une postion à peine différentes. Il faut précisier que les mesures sur la position 5 ont été effectuées au même moment tant pour les 40 que les 5 mesures. C'est pour cette raison que la détection de cette position est très bonne.

Cela montre une précision de 43.33% ce qui correspond à 13 positions correctement reconnues sur 30 alors que les résultats pour les positions convergées donnent 83.33% qui correspond à une reconnaissance correcte de 25 position sur 30.

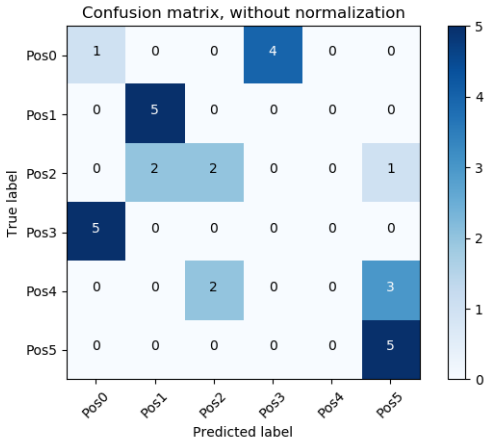


Figure 4.11 – Matrice de confusion obtenue en utilisant les poitns SxT et la caractérisitque de position

Meilleur résultat obtenu

Ce chapitre va présenter le meilleur résultat qui a été obtenu ainsi que les caractéristiques qui ont été utilisées. Pour y arriver les caractéristiques suivantes ont été testée de manière individuelle :

1. Toutes les données "ranging" (raw)
2. Donnée ranfing d'un canal (raw)
3. La moyenne des cannaux
4. La variance sur les cannaux
5. Déviation standard sur les cannaux
6. Quantile effectué sur les cannaux
7. Médiane effectuée sur les cannaux
8. Covariance des cannaux

Ensuite, plusieurs association ont été faites entre ses différents test et le meilleur des résultats est obtenu en associant la valeur de la médiane, de la variance, du quantile et des données de "ranging". A noter que pour avoir le meilleur résultat, il a été nécessaire de prendre en compte uniquement la deuxième mesure de ranging sur chaque esclave. Lorsque les différentes caractéristiques sont affichée, il est à noter que l'amplitude des valeurs n'est pas du même ordre et par conséquent pour tenter de les égaliser, la médiane a été multipliée par cinq, le quantile par six et la variance a été divisée par quatre. Cette façon de faire à encore amélioré les résultats pour obtenir ceux présenté ci-dessous

La Figure 4.12 présente le résultat obtenu lorsque l'entrainement est fait avec 32 des 40 mesures sur une même position et testé avec les 8 restants. Cela donne de très bonnes précisions qui se situent entre 83% et 98%. Les position sont majortiairement bien reconnues. La variation de pourcentage pour la reconnaissance vient du fait qu'à chaque lancement, le programme exclu 8 autres mesures.

La matrice de confusion représentée est donnée avec des mesures qui ont été faites sur une même point sans bouger l'espion. La précision dans ce cas est de 97.92% avec un F1-score macro de 0.9791 et un F1-score micro de 0.9792.

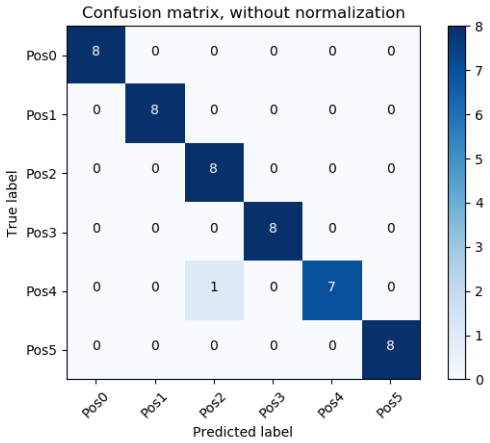


Figure 4.12 – Matrice de confusion obtenue pour la meilleures solution en utilisant les données des point Sx

Maintenant, afin de valider le système 5 points suplémentaires par postion ont été pris durant une autre période de la journée et en ayant propablement une postion à peine différentes. La présentation des précédents résultat montre de très mauvais résultats concernant ce test mais après le travail effectué et la recherches des meilleures caractéristiques les résultas sont satisfaisants et encourageant. Il faut précisier que les mesures sur la position 5 ont été effectuées au même moment tant pour les 40 que les 5 mesures. C'est pour cette raison que la détection de cette position est très bonne.

Cela montre une précision de 73.33% ce qui correspond à 22 positions correctement reconnue sur 30.

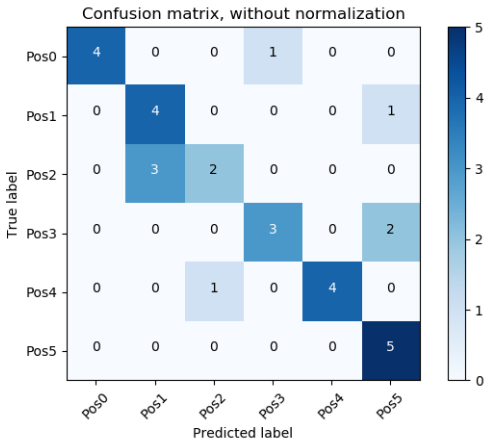


Figure 4.13 – Matrice de confusion pour la meilleures solution en utilisant les point de tests SxT

Un dernier test a été réalisé afin de voir si les positions Tx sont bien reconnue par rapport à leur plus proche voisin. La Figure 4.14 montre les résultats obtenus lorsque les positions Tx sont utilisées. Les points T1/2/7/8 correspondent à la position S0, T3/4/9 correspondent à la position S1, T5 correspond à la position S2, T0/6 correspondent à la position S3 et finalement, T10/11/12 correspondent à la position S4. Comme attendu les résultats pour cette classification ne sont pas bons est pas utilisable. La précision obtenue est de 31.25% ce qui représente une détection correcte de 20 positions sur 64. Pour obtenir ce résultat on essaie de classifier les positions Tx selon la position entrainée la plus proche.

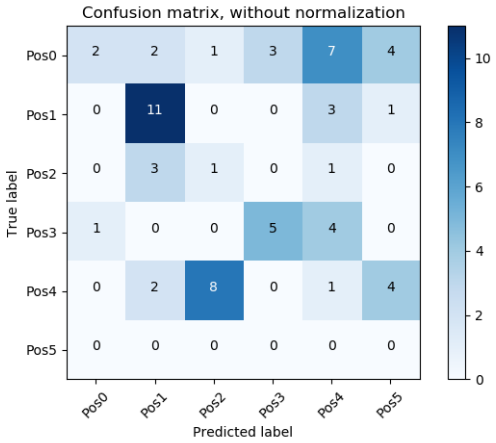


Figure 4.14 – Matrice de confusion pour les valeurs de la position convergée avec des points de tests positionnés autour des points entrainés

4.4.2 Résultats en utilisant les données de position convergées

Ce chapitre va présenter les résultats qui ont été obtenus à partir des approximation de position fournies par le programme de mesures. Dans cette partie uniquement ces valeurs seront utilisée. Afin d'avoir une meilleure vue de ce que représente ces positions, la Figure 4.15 montre le positionnement des points calculés par rapport à la position réelle de l'espion. Les gros points de couleurs représentent la position réelle et les petits points de couleurs représentent les positions calculées. La croix noir représente la position du master.

La Figure 4.15 met en évidence un phénomène qui tend à faire correspondre les points proches du maitre sur les coordonnées du maitre. Cet effet qui se trouve dans le laboratoire où les mesures ont été effectuées complique la reconnaissance de position. De par cette représentation, il est possible de voir à l'oeil nu la séparation des points par contre, il est rapidement possible de se rendre compte que si on ajoute des points il sera de plus en plus difficile à différencier les emplacements de base de l'espion.

Ci-dessous, une énumeration des couleurs et de la position associée :

1. Couleur verte (S0) : x = 21.30m / y = 5.60m
2. Couleur jaune (S1) : x = 17.30m / y = 5.60m
3. Ccouleur bleue (S2) : x = 17.30m / y = 2.10m
4. Couleur rose (S3) : x = 21.30m / y = 3.10m
5. Couleur rouge (S4) : x = 13.80m / y = 5.60m
6. Couleur noir (S5) : x = 13.80m / y = 3.10m

La Figure 4.16 représente le résultat obtenu lorsque l'entrainement est fait avec 32 des 40 mesures sur une même position et testé avec les 8 restants. Cela donne de très bonnes précisions qui se situent entre 81% et 89%. Comme attendu, les positions proches du maitre (S4 et S5) se confondent et pas conséquent ne sont pas correctement reconnues. Les autres position sont majortiairement bien reconnu à l'exception de quelques mesures. La variation de pourcentage pour la reconnaissance vient du faire qu'à chaque lancemnt, le programme exclu 8 autres mesures.

Afin de tester l'algorithme fixé, il est entrainé avec les 40 mesures effectuée sur chaque position de l'espion. Ces 40 mesures sont celles utilisée pour définir les meilleurs paramètres et obtenir les meilleurs résultats comme présenté dans la Figure 4.15. La Figure 4.17 présente le resultat final obtenu avec les cinq données de test qui avaient été effectué à la même position que les quarante de l'entrainement.

La précision obtenue pour ce test est de 83.33% ce qui correspond à une reconnaissance correcte de 25 position sur 30. Les positions mal reconnue sont sans surprise les positions S4 et S5.

La Figure 4.18 montre les résultats obtenus lorsque les positions Tx sont utilisées. Les points T1/2/7/8 correspondent à la position S0, T3/4/9 correspondent à la position S1, T5 correspond à la position S2, T0/6

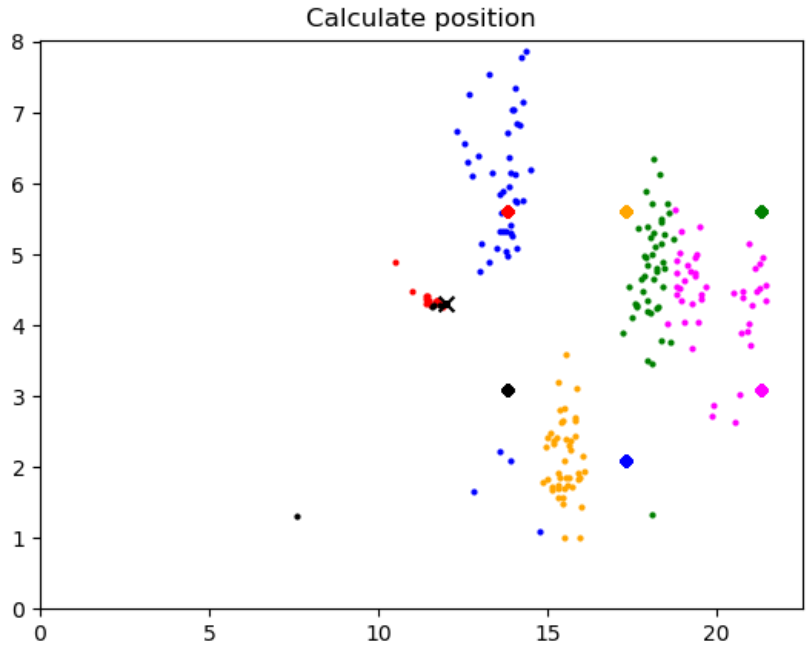


Figure 4.15 – Représentation des points convergés par rapport aux positions réelles

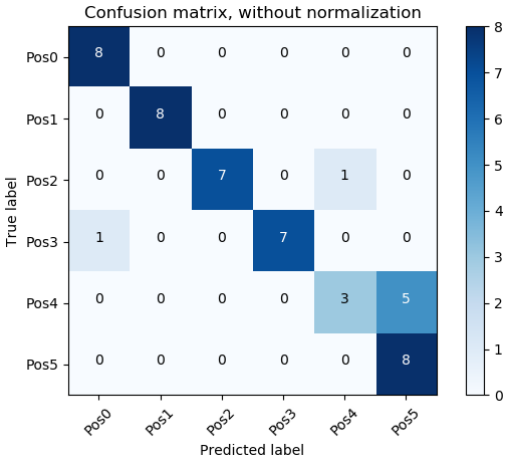


Figure 4.16 – Matrice de confusion pour les valeurs de la position convergée avec des points positionnés à la même place

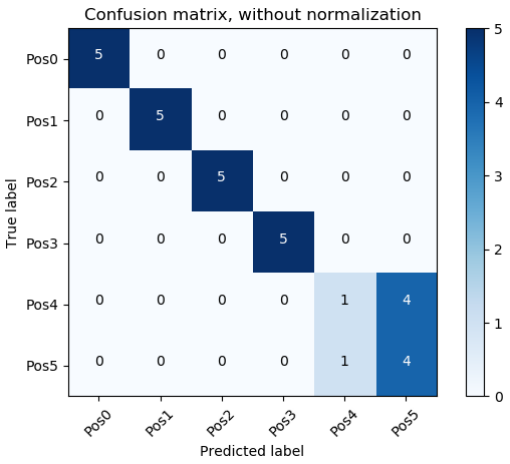


Figure 4.17 – Matrice de confusion pour les valeurs de la position convergée avec des points de tests positionnés à la même place

correspondent à la position S3 et finalement, T10/11/12 correspondent à la position S4. Comme attendu les résultats pour cette classification n'est pas bonne est pas utilisable. La précision obtenue est de 21.67% ce qui représente une détection correcte de 13 position sur 60. Pour obtenir se résultat on essaie de classifier les positions Tx selon la position entrainée la plus proche.

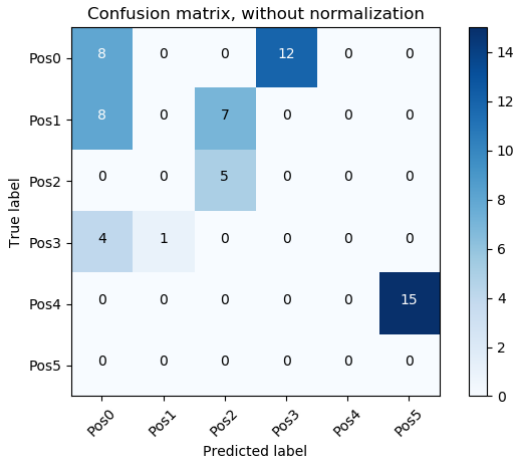


Figure 4.18 – Matrice de confusion pour les valeurs de la position convergée avec des points de tests positionnés autour des points entrainés

4.5 Résumé des résultats

La Figure 4.19 montre un résumé des résultats présentés dans ce rapport. La solution la plus optimiste a été mise en vert. Ce n'est pas celle qui donne les meilleurs résultats au niveau des points SxT mais malgré cela c'est elle qui offre la plus grande liberté de traitements et d'améliorations.

La solution qui utilise uniquement la position convergée donne effectivement de bons résultats mais si le nombre de points est augmenté ils vont commencé de se mélanger et il ne sera plus possible de les différencier. Les positions proches du maitre seront toujours confondue alors qu'avec la solution d'utiliser les données brutes ces positions sont différenciable. Lorsque les données brutes de "ranging" sont utilisées il est possible d'utiliser des traitements sur ces mesures comme la moyenne, la déviation standard, la

variance, etc... chose qui n'est pas possible lorsqu'on possède uniquement l'information des coordonnées du point x et y.

Caractéristiques utilisées	Précision obtenue sur les mesures à des positions 100% identiques (Sx)	Précision obtenue sur les positions identique mais un autre jour (SxT)	Précision obtenue sur les positions entourant les positions d'entrainement (Tx)
Toutes les données "ranging"	97.92%	46.67%	-
Données de "ranging" 2ème mesure	97.92%	53.33%	-
Données de positions non convergées	72.92%	43.33%	-
Données de "ranging" 2ème mesure / médiane / quantile / variance	97.92%	73.33%	31.25%
Données de positions convergées	85%	83.33%	21.67%

Figure 4.19 – Résumé des différents résultats obtenus qui sont présentés dans ce rapport

5 Problèmes rencontrés et améliorations

Ce chapitre va traiter des problèmes qui ont été rencontré tout au long de ce travaille d'approfondissement ainsi que des améliorations possibles.

Ce travaille est basé sur une thèse de Master que Michael Mueller avait réalisé [Mic17]. Durant ce travail il avait mis en place un programme permettant de calculer une position dans un local intérieur. Réutiliser ce programme devait permettre d'acquérir facilement des mesures mais cela n'a pas été le cas. Dans un premier temps il a été nécessaire de modifier le programme car le système de prise de mesures n'était pas adapté. Cela dans le sens qu'il effectuait des mesures en permanence en stockant les résultats et en recalculant à chaque fois une position par rapport à toutes les mesures.

Ce qui a été problématique et la stabilité du système qui à tout moment s'arrête de fonctionner et ne donne plus de résultat utilisable. Les esclaves cessaient de fonctionner de temps en temps et cela faisait qu'une mesure prenait un temps conséquent. Il a aussi été remarqué que lorsque le système fonctionnait durant de longues heures d'affilées les bug étaient de plus en plus nombreux par finir de n'être plus utilisable.

De plus, lorsque les mesures de positions convergées ont été réalisée afin d'obtenir une série de mesure sur une position (40 mesures sur les 40 canaux) il était nécessaire d'avoir une demi journée et de surveiller le système pour que un des esclave ne s'éteigne pas.

Au niveau des améliorations, il serait nécessaire de travailler sur le programme de prise de mesures afin de comprendre pourquoi le système s'arrête parfois. Car actuellement, la prise est automatique mais si un des esclaves s'arrête, il faut le redémarrer et cela ne peut pas être fait de manière automatique.

Un autre point serait de travailler sur un premier tris des mesures afin d'utiliser uniquement les mesures qui donnes de bons résultats et supprimer les autres. Cela permettrait d'avoir de meilleurs résultats finaux.

Concernant la compréhension du système complet il serait nécessaire de faire les mesures dans un environnement maitrisé où les réflexions sont moindre ce qui n'est pas du tout le cas dans le laboratoire. La quantité de mesures prises doit également être augmentée afin de pouvoir poursuivre le travail et se pencher sur une analyse par régression.

Il serait également nécessaire de travailler sur le positionnement des esclaves et du maitre. Peut -être qu'il serait intéressant d'ajouter un second maitre et de les placer au extrémité plutôt qu'au centre du local.

Finalement, l'intégrations de la mesure du RSSI pourrait être un candidat très intéressant pour compléter l'analyse.

6 Conclusion

Bibliographie

[LLI] Mainetti Luca, Patrono Luigi et Sergi Ilaria. *A Survey on Indoor Positioning Systems*. Rapp. tech. University of Salento Lecce, ITALY (cf. p. 1).

[FP17] Bernat Carbones Fargas et Martin Nordal Petersen. « GPS-free Geolocation using LoRa in Low-Power WANs ». In : *Proceedings of 2017 Global Internet of Things Summit (GIoTS)* (2017) (cf. p. 2, 4, 7).

[Ped+11] F. Pedregosa et al. « Scikit-learn : Machine Learning in Python ». In : *Journal of Machine Learning Research* 12 (2011), p. 2825-2830 (cf. p. 3, 6).

[KN17] Khuong et Nguyen. « A performance guaranteed indoor positioning system using conformal prediction and the WiFi signal strength ». In : *Journal of Information and Telecommunication, 1 :1, 41-65, DOI : 10.1080/24751839.2017.1295659* (2017). URL : https://doi.org/10.1080/24751839.2017.1295659 (cf. p. 4, 11-14).

[JR14] Quinlan J. R. « C4. 5 : programs for machine learning ». In : *Elsevier* (2014) (cf. p. 4).

[GHP95] John G. H. et Langley P. « Estimating Continuous Distributions in Bayesian Classifiers ». In : *11th Conference on Uncertainty in Artificial Intelligence, pp., 338-345* (1995) (cf. p. 4).

[CS13] Anuradha C. et Dhall S. « Software Defect Prediction Using Supervised Learning Algorithm and Unsupervised Learning Algorithm ». In : (2013) (cf. p. 4).

[WA13] Yotsawat W. et Srivihok A. « Inbound tourists segmentation with combined algorithms using K-Means and Decision Tree ». In : *10th International Joint Conference on Computer Science and Software Engineering (JCSSE), pp.189-194* (2013) (cf. p. 4).

[SS14] Ureerat S. et P. Singsri. « The classifier model for prediction quail gender after birth based on external factors of quail egg ». In : *IEEE 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)* (2014) (cf. p. 4).

[DL07] Yang D. et Jin-lin L. « Research on personal credit evaluation model based on bayesian network and association rules ». In : *International Conference on Wireless Communications, Networking and Mobile Computing* (2007) (cf. p. 5).

[DWDMK91] Aha D. W., Kibler D. et Albert M. K. « Instance-based learning algorithms ». In : *Machine Learning, vol. 6, pp., 37-66* (1991) (cf. p. 5).

[CAG13] Shah C. et Jivani A. G. « Comparison of data mining classification algorithms for breast cancer prediction ». In : *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)* (2013) (cf. p. 5).

[J.98] Platt J. « Fast Training of Support Vector Machines using Sequential Minimal Optimization ». In : *Advances in Kernel Methods - Support Vector Learning* (1998) (cf. p. 5).

[PH14] Niken P. et Ohwada H. « Applicability of machine-learning techniques in predicting customer defection ». In : *IEEE 2014 International Symposium on Technology Management and Emerging Technologies (ISTMET)* (2014) (cf. p. 5).

[SMKN13] Obaidullah S. M., Roy K. et Das N. « Comparison of different classifiers for script identification from handwritten document ». In : *IEEE International Conference on Signal Processing, Computing and Control (ISPCC), pp.1-6* (2013) (cf. p. 5).

[YRE96] Freund Y. et Schapire R. E. « Experiments with a new boosting algorithm ». In : *3th International Conference on Machine Learning, San Francisco, pp. 148-156* (1996) (cf. p. 5).

[RRE13] Shams R. et Mercer R. E. « Classifying Spam Emails Using Text and Readability Features ». In : *IEEE 13th International Conference on Data Mining (ICDM, pp. 657-666* (2013) (cf. p. 5).

[SOLISP10] Sharif S. O., Kuncheva L. I. et Mansoor S. P. « Classifying encryption algorithms using pattern recognition techniques ». In : *IEEE International Conference on Information Theory and Information Security (ICITIS), , pp. 1168-1172* (2010) (cf. p. 5).

[L.96] Breiman L. « Bagging predictors ». In : *Machine Learning. vol. 24, no. 2, pp.123-140* (1996) (cf. p. 5, 6).

[Hen+12] Wymeersch Henk et al. *A Machine Learning Approach to Ranging Error Mitigation for UWB Localization*. Rapp. tech. Belgian American Education Foundation, the Charles Stark Draper Laboratory Robust Distributed Sensor Networks Program, the Office of Naval Research Young Investigator Award N00014-03-1-0489, the National Science Foundation under Grants ANI-0335256 and ECCS-0636519., 2012 (cf. p. 6, 10, 11).

[CC06] Rasmussen C. et Williams C. « Gaussian Processes for Machine Learning ». In : *Springer* (2006) (cf. p. 6).

[VT94] Barnett V. et Lewis T. « Outliers in Statistical Data ». In : *3rd ed. Wiley Series in Probability and Mathematical Statistics* (1994). URL : https://doi.org/10.1080/24751839.2017.1295659 (cf. p. 7).

[Esd] « lien concernant les ESD ». In : (). URL : https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h3.htm (cf. p. 7).

[Sin+15] Bozkurt Sinem et al. *A Comparative Study on Machine Learning algorithms for Indoor Positioning*. Rapp. tech. This work is supported by The Scientific et Technological Research Council of Turkey (TUBITAK) under grant number 1130024, 2015 (cf. p. 7-10).

[Wika] « lien concernant la regression non paramétrique ». In : (). URL : https://fr.wikipedia.org/wiki/R%C3%A9gression_non_param%C3%A9trique (cf. p. 10).

[Mic] « lien concernant le concours Microsoft ». In : (). URL : https://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2014/ (cf. p. 12).

[Mic17] Mueller Michael. « Système de localisation et de tracabilité des objets communicant basse consommation ». Mém. de mast. HES-SO Master, 2017 (cf. p. 15, 16, 41).

[Wikc] « lien concernant NumPy ». In : (). URL : https://fr.wikipedia.org/wiki/NumPy (cf. p. 26).

[Wikd] « lien concernant Pandas ». In : (). URL : https://fr.wikipedia.org/wiki/Pandas (cf. p. 26).

[Dat] « lien concernant le score F1 ». In : (). URL : https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin/16001 (cf. p. 30).

[Wikb] « lien concernant le score F1 ». In : (). URL : https://en.wikipedia.org/wiki/F1_score, https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel.