Projet DOMOTIQUE – Rapport

Clément Hérouard Marie-Morgane Paumard

***Résumé*—Dans le cadre de la domotique, la détection d'activité permet de gérer diverses applications de l'habitat. Nous nous concentrons sur une approche récente utilisant la théorie des fonctions de croyances. Dans cette approche, les capteurs renvoient une fonction de masse et doivent pour cela être calibrés. Les fonctions de masse représentent les connaissances abstraites que l'on extrait des données d'un capteur. Si la calibration est de mauvaise qualité, ces connaissances seront erronées et fausseront donc les décisions d'action dans l'habitat.**

Nous avons étudié un processus de calibration automatique, basé sur l'algorithme de classification k-Nearest-Neighbors (k-NN). Ce dernier repose sur plusieurs paramètres. Nous avons donc fait varier ces divers paramètres pour améliorer la qualité de la calibration. Nous cherchons des interactions entre ces paramètres et à les expliquer.

Ce document se veut une étude d'une méthode de calibration automatique des fonctions de masses à l'usage de la domotique. Nous avons étudié l'algorithme k-NN et proposons, dans ce papier, quelques méthodes de paramétrage intéressantes, notamment une basé sur l'algorithme de descente de gradient, ou l'influence d'un pré-traitement des données brutes.

I. Introduction

La maison intelligente permet d'améliorer le confort et la sécurité de ses occupants, en s'adaptant aux comportements des utilisateurs et de l'environnement. Pour éviter des approches basées sur les scénarios, trop primitifs et peu adaptés, nous nous intéresserons à la reconnaissance de contexte. Dey [1] décrit le contexte comme « toute information permettant de caractériser la situation d'une entité ». Ainsi, si le système doit déterminer si quelqu'un cuisine ou fait la vaisselle, le contexte utilisé pourrait être la consommation d'eau ou la température du four.

Certaines activités nécessitent l'usage de capteurs hétéroclites. Ainsi, pour détecter la présence d'un individu, il est envisageable d'utiliser des capteurs de mouvements, de CO2 ou de décibels. Cette diversité de capteur fournit une masse de données particulièrement hétérogène. Nous calibrons les capteurs puis filtrons les données qu'ils retournent, nous les trions et les fusionnons, de manière à les rendre exploitables.

L'équipe TACOMA [2] a déjà proposé une ligne directrice prometteuse pour fusionner des données retournées par plusieurs capteurs. Pour notre projet, nous étudierons cette méthode pour automatiser la calibration des capteurs. En effet, la calibration manuelle des capteurs est trop longue à effectuer pour être réalisée sur tous les capteurs. Nous considérerons plus précisément l'algorithme des k-nearest-neigbhors (k-NN), en étudiant l'influence de chacun de ses paramètres sur la qualité de notre calibration. Nous en déduirons des heuristiques pour la calibration automatique.

Ce rapport est organisé comme suit : en section II, nous proposons quelques outils utilisés pour la reconnaissance de contexte. Puis, après avoir présenté et justifié les expériences que nous avons choisi de mener sur la détection de présence dans une pièce, nous détaillons les méthodes que nous appliquons pour déterminer un bon paramétrage de k-NN. En section V, nous les comparons et montrons que la méthode du gradient est optimale. La section suivante avance quelques critiques sur nos expériences.

II. Outils

L'équipe TACOMA développe un système permettant de déterminer un contexte, c'est-à-dire une activité en cours, à partir de données de plusieurs capteurs. Nous présentons ici quelques-uns des outils utilisés pour la reconnaissance de contexte.

A. Détermination du contexte

La chaîne de traitement des données se décompose en plusieurs étapes.

1) Chaque capteur envoie des données brutes.
2) Les données brutes sont pré-traitées pour calculer une variable intéressante, comme détaillé en partie III-B.
3) Ces données sont utilisées par une **fonction génératrice** dont le rôle est de créer des informations exploitables, en faisant abstraction des données. Cette fonction prend en entrée des données de capteur et renvoie un objet appelé **fonction de masse** et qui décrit la croyance du capteur de manière abstraite.
4) Chaque capteur permet d'obtenir une fonction de masse. Les fonctions de masse de mêmes natures (i.e., associées à un même contexte, comme par exemple la détection de présence) sont fusionnées pour déterminer un contexte global.

Il existe plusieurs méthodes pour fusionner des fonctions de masse. Ces fonctions ont été étudiés par Pietropaolli [2] dans sa thèse et ses travaux proposent deux méthodes : celle de Dempster [7] et celle de Dubois et Prade. Nous avons privilégié cette dernière.

Cette méthode nécessite d'avoir une fonction génératrice par capteur et par contexte à reconnaître.Cela signifie qu'il faut pouvoir associer une fonction de masse à chaque valeur

possible fournie par le capteur. Actuellement, pour des capteurs de mouvement, cette fonction est créée de manière empirique. Néanmoins, cette méthode n'est pas envisageable en pratique : pour effectuer une calibration chez des particuliers, il serait nécessaire de disposer de techniciens maîtrisant les fonctions de masse, pour chaque capteur d'un bâtiment. De plus, une telle calibration est nécessaire à l'installation du système, mais pourra également devoir se faire en cas de vieillissement d'un capteur ou de déplacement du mobilier.

Pour que le système soit utilisable, il est donc nécessaire de pouvoir calibrer automatiquement nos capteurs. Nous allons ainsi proposer des méthodes permettant d'obtenir de bonnes fonctions génératrices pour nos capteurs. Une mauvaise fonction génératrice engendrera une fonction de masse qui représentera des connaissances erronées. Il convient donc de définir un critère pour caractériser les bonnes fonctions de masse.

B. Fonctions de masse et fonctions génératrices

Premièrement, définissons un ensemble d'attributs de contexte possibles. Cet ensemble est nommé **cadre de discernement**. Dans notre cas d'étude, nous distinguerons deux possibilités : présence ou absence de personnes dans la pièce. Une fonction de masse sur le cadre de discernement Ω est une fonction f allant des parties non vides de Ω vers le segment $[0, 1]$ vérifiant : $\sum_{A \subseteq \Omega} f(A) = 1$. Ces fonctions expriment une probabilité sur les parties de Ω, et non sur les éléments de Ω.

Les fonctions de masse permettent alors d'exprimer une information incertaine, ce qui est idéal pour nos capteurs. Cela s'effectue en mettant du poids sur Ω, que l'on appelle **incertitude totale**. Ω est la réunion de tous les attributs de contexte, donc mettre du poids sur cet ensemble signifie que l'on ne peut en discerner aucun, et donc représente bien l'incertitude d'un capteur. De même, les fonctions de masse permettent d'exprimer l'imprécision d'un capteur. Par exemple, si nous avons plusieurs cas possible, un capteur peut aider à éliminer certains de ces cas, sans trancher entre les autres. Pour finir, il convient de rappeler qu'il existe des méthodes pour fusionner des fonctions de masse, et ainsi additionner les connaissances apportées par chacune.

Dans le cas qui nous intéresse, la détection de présence, une fonction de masse donne donc simplement un pourcentage de croyance en la présence d'une personne, un autre pour l'absence, et un dernier caractérisant l'incertitude. Par exemple, nous pouvons avoir pour fonction de masse : présence = 0%, absence = 20%, ignorance = 80%, le capteur semble indiquer l'absence, mais de manière très incertaine.

Pour évaluer la qualité d'une fonction génératrice, il faut utiliser un jeu de données de test. Il s'agit d'une liste de valeurs pré-traitées et étiquetées par l'attribut de contexte pour lequel la valeur a été obtenue. La méthode d'acquisition de ce jeu de données sera détaillée en partie III. Ainsi, nous avons testé notre fonction génératrice sur les données du jeux de test. Pour chaque donnée, elle a produit une fonction de masse, que nous avons comparé avec une autre fonction de masse dont tout le poids est concentré sur l'attribut de contexte qui étiquette la fonction. Cette dernière fonction représente la réalité, car nous avons mémorisé le contexte lors de l'acquisition. Nous avons alors utilisé la distance décrite dans l'article [8] pour calculer la distance entre les deux fonctions de masse. Cela permet en quelque sorte de calculer une distance en notre fonction de masse et le cas réel. Cette distance est comprise entre 0 et 1. Nous disposons donc d'une distance par valeur de jeu de test. Pour obtenir un critère de qualité de la fonction génératrice, nous calculons la distance moyenne sur le jeu de test. Ainsi, ce test permet d'évaluer les erreurs commises par la fonction génératrice : plus cette erreur est faible, meilleure est la fonction génératrice. Nous considérerons qu'une fonction génératrice est bonne si l'erreur est inférieure à 0.15%.

C. Algorithme des k-Nearest Neighbors

Pour calculer la fonction génératrice, nous utilisons un algorithme basé sur l'algorithme de clustering k-Nearest Neighbors (k-NN). Cet algorithme est dit supervisé, c'est-à-dire que l'utilisateur doit fournir certaines informations lors de la mise en place pour guider l'apprentissage. L'utilisateur produira ainsi un jeu de données de référence dont s'inspirera l'algorithme pour étiqueter les données qu'il recevra ensuite. Utiliser un algorithme supervisé permet de disposer d'un système fonctionnel dès la fin de la calibration, là où un algorithme non-supervisé commencerait par faire beaucoup d'erreur après l'installation.

Pour créer la fonction génératrice, nous utilisons donc une liste de valeurs de capteurs, étiquetées par le contexte associé. L'idée de l'algorithme est d'apprendre ces points et pour calculer la fonction de masse à partir d'une valeur ; il suffit d'observer les points appris à proximité de la valeur en question. On connaît les contextes de ces points, donc on utilise l'heuristique que le contexte associé à la valeur est probablement le même que celui de ces voisins.

Formellement, pour calculer la fonction de masse associée à une valeur v, il faut chercher les k points de la liste d'apprentissage les plus proches de v_0. k est un paramètre fixé initialement. Ensuite, pour chacune des k valeur sélectionnée v, étiquetée par le contexte C et à une distance de v_0 égale à d, nous créons la fonction de masse f_p suivante :

$$f(C) = \alpha \cdot e^{-\beta \cdot d}$$
$$f(\Omega) = 1 - \alpha \cdot e^{-\beta \cdot d}.$$

α est un paramètre compris entre 0 et 1 et β est un paramètre quelconque. Cette fonction de masse représente donc une fonction de masse indiquant une croyance dans le contexte

C. Plus forte si v_0 et v sont proche et si α et grand. Nous avons ainsi k fonctions de masse. Nous allons les fusionner pour obtenir la fonction de masse associée à la valeur v_0.

Intuitivement, pour trouver la fonction de masse associée à une valeur, nous cherchons les k valeurs apprises les plus proches. Ainsi l'on accorde plus de crédit aux contextes associés aux points les plus proches. La confiance maximale que l'on accorde à un seul point est égale au paramètre α.

Nous allons donc chercher à trouver les paramètres (k, α) qui permettent d'obtenir la fonction génératrice de la meilleure qualité possible. Nous allons donc mesurer l'erreur de ces fonctions à l'aide d'un jeux de test comme vu en II-B.

III. Mesures et pré-traitement

Nos expériences avaient pour objectif de produire deux jeux de données : un jeu d'apprentissage pour entraîner notre modèle, et un jeu de test pour vérifier ce qu'il a appris. Nous étudions la détection absence/présence, et disposions pour cela de capteurs de mouvement ; il nous fallait déterminer combien nous allions en utiliser et comment les placer. Nous devions aussi choisir la durée de mesure et quel taux d'activité serait le plus intéressant.

A. Mesures

Nous avons utilisé deux capteurs de mouvement de même marque de manière à pouvoir comparer nos résultats. Nous avons effectué trois jeux d'absence, durant lesquels il n'y avait personne dans la pièce, et trois jeux de présence durant lesquels une personne restait dans le champ de mesure, en bougeant peu. Ces jeux contiennent 300 points chacun.

L'intérêt d'utiliser des capteurs de même modèle est de déterminer s'il y a une différence de calibration usine entre ces capteurs.

B. Pré-traitement des données brutes

Les données brutes du capteur sont comprises entre 0 et 1024 et l'absence de mouvement est caractérisée par des valeur proches de 512. Nous effectuons un premier pré-traitement pour centrer nos valeurs. Ainsi l'absence est désormais caractérisée par des valeurs très faibles, et la présence par des valeurs plus hautes.

Néanmoins, d'autre données peuvent être utiles, comme la variation de cette valeur. Nous avons donc calculé les variables suivantes : la dérivée de la valeur, la moyenne, le minimum et le maximum sur les trois dernières valeurs.

L'algorithme k-NN ne nous limite pas aux variables à une seule dimension. Nous avons dons essayé d'utiliser plusieurs variables en même temps, comme par exemple le couple valeur-dérivée du capteur.

Si le multi-traitement présente un défaut en terme de temps de calcul (30%) fois plus lent, nous avions supposé qu'il augmenterait sensiblement la précision du couple (k, α) obtenu. L'expérimentation nous permettra de savoir si cette augmentation en vaut la peine.

IV. Méthodes

Nous allons proposer deux manières d'obtenir un bon couple (k, α), c'est à dire dont l'erreur est faible.

A. Recherche exhaustive du meilleur couple (k, α)

La première méthode que nous avons mise en œuvre est le parcours exhaustif de toutes les combinaisons de k et α. Cette méthode est précise car elle nous permettra d'obtenir le meilleur couple (k, α). Néanmoins, elle est coûteuse en temps de calcul puisque l'on teste tous les couple (k, α) possibles. Elle n'est donc pas acceptable en terme de temps de calcul, notamment puisque l'équipe TACOMA souhaite utiliser ces algorithmes sur un Raspberry Pi, qui est un ordinateur très peu puissant.

B. Descente de gradient

Face à notre problème d'optimisation, nous avons implémenté la méthode de descente de gradient. Cette méthode permet de minimiser la fonction d'erreur en optimisant notre temps de calcul et donc de s'adapter à nos contraintes matérielles.

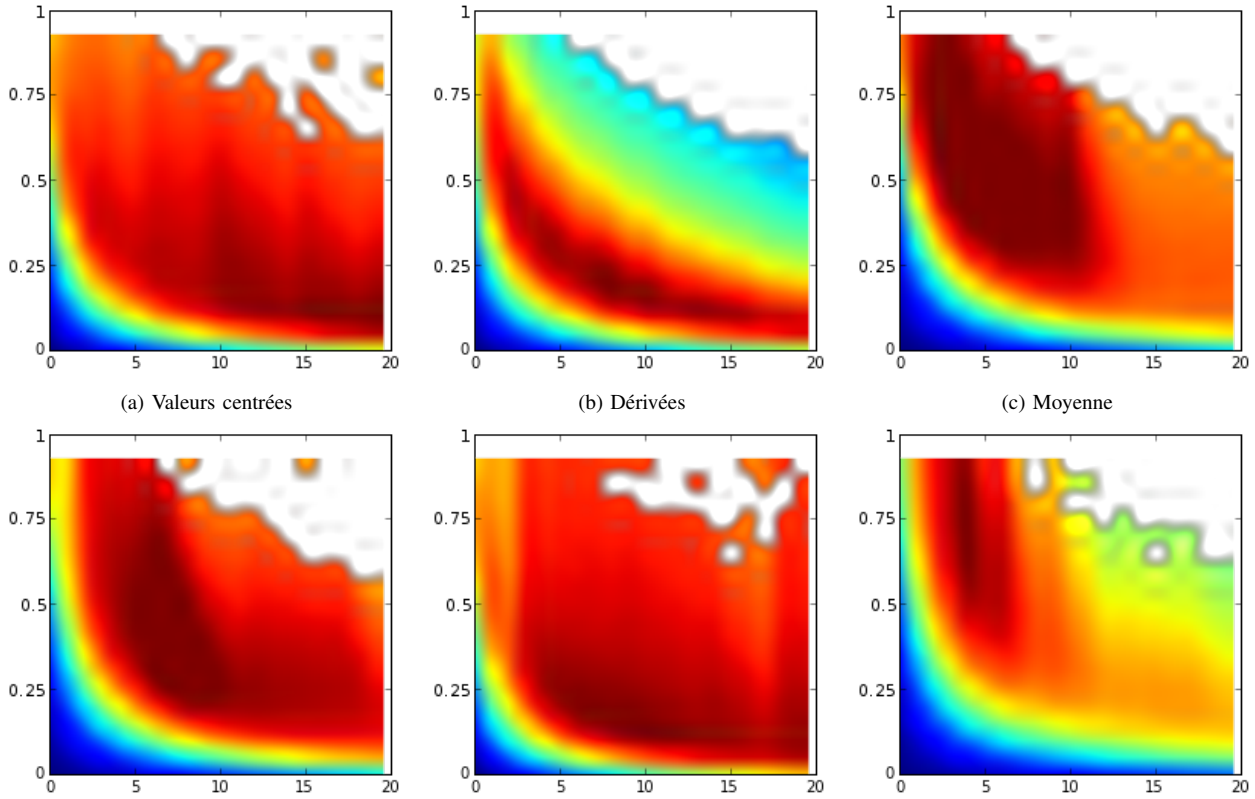
La descente de gradient consiste à partir d'un couple initial (k, α) et faire évoluer ce couple jusqu'à converger vers un bon couple. Pour cela, nous calculons le gradient et nous déplaçons notre couple (k, α) selon la pente la plus forte. Ainsi, il s'agit d'une méthode gloutonne qui améliore le couple à chaque itération jusqu'à convergence. La solution obtenue est donc un minimum local ; en l'appliquant nous n'obtiendrons donc pas nécessairement le meilleur couple (k, α). Nous verrons toutefois en partie V que celui-ci suffira.

Le choix primordial pour appliquer cette méthode est de sélectionner un point initial probant, car les mimima trouvés dépendent uniquement de celui-ci. Nous avons donc pris comme couple initial le couple (1, 0). Nous partons ainsi d'un k et d'un α les plus petits possible. Ce choix est induit par l'observation des carte de température obtenues grâce à la méthode exhaustive. Nous reviendrons sur ce choix en partie V.

V. Résultats

Après avoir effectué plusieurs jeux de mesure, nous avons appliqué des pré-traitements (III-B) et des méthodes (partie IV) pour comparer les couples (k, α) obtenus, les taux d'erreur et les temps de calculs. Les temps de calculs ont été obtenus avec un processeur Intel i7 (200GFLOPS) et non pas un Raspberry Pi (0.04GFLOPS).

En figure 1, nous proposons diverses cartes de températures, pour la méthode exhaustive. Certaines cartes présentent des erreurs faibles pour de très nombreuses valeurs,



(a) Valeurs centrées (b) Dérivées (c) Moyenne

(d) Maximum (e) Minimum (f) Maximum et minimum

FIGURE 1: Cartes de températures représentant l'erreur : plus le rouge est sombre plus l'erreur est faible. Inversement, le bleu signifie que l'erreur est très grande. L'axe des ordonnées représente α et l'axe des abscisses, k.

Méthodes	Exhaustive 1D	Gradient 1D	Exhaustive 2D	Gradient 2D
Meilleur(s) traitement(s)	max	max	max, min	max, min
Taux d'erreur	0.063	0.073	0.060	0.082
Temps de calcul (secondes)	18.7	6.6	40.1	12.4
Valeur de k obtenue	9	18	5	20
Valeur de α obtenue	0.4	0.162	0.75	0.172

TABLE I: Extrait des taux d'erreurs et temps de calculs obtenus pour le meilleur traitement de chaque méthode (recherche exhaustive ou descente de gradient, nombre de traitements effectués).

comme la moyenne, tandis que d'autres sont beaucoup plus sélectives, comme la dérivée. Ces cartes proposent des couples (k, α) efficaces disjoints pour des taux d'erreurs proches. Ainsi la carte (1c) nous donne que le meilleur couple est (4, 0.6) tandis que la (1e) retourne (10, 0.55) ; les erreurs sont respectivement 0.60% et 0.55%. La table II en annexes détaille les valeurs obtenues.

La table II, en annexe, permet également de montrer qu'employer une variable à deux dimensions grâce au prétraitements n'est pas nécessairement optimal, bien que c'est la combinaison maximum et minimum qui nous permet d'aboutir au meilleur résultat.

La plupart des cartes de températures présentent des minima locaux. Ils sont toutefois suffisamment proches du minimum global pour que l'approche de la méthode du gradient soit raisonnable, comme souligné par la table III, en annexe. Dans la majorité des cas, la différence d'erreur est inférieure à 0.02%. En revanche, la différence de temps de calcul n'est pas négligeable : les calculs sont plus de trois fois plus rapides avec la méthode du gradient. Nous avons

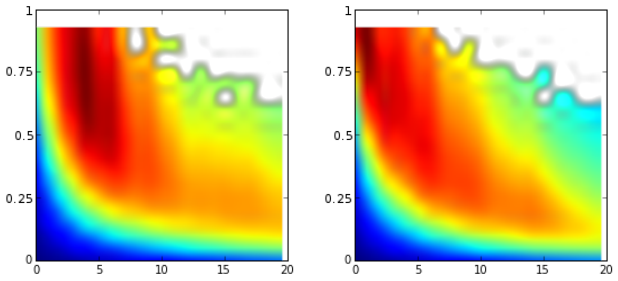
par ailleurs choisi comme point initial de la méthode du gradient le point (1, 0) car les cartes de température semble toute présenter une "hyperbole de bon couple". Pour des k et α plus élevés, il y a des problème de précision numérique important (causant l'absence de ces points sur la carte). Le point (1, 0) nous a donc paru pertinent pour converger vers cette hyperbole, plutôt qu'un point aléatoire.

VI. Discussion

Sans contrainte de temps, les meilleurs résultats sont obtenus avec la méthode exhaustive et le couple de traitements maximum et minimum : $k = 5$, $\alpha = 0.75$ et l'erreur vaut 0.06%. En prenant le temps en compte, la meilleure méthode est celle du gradient avec pour seul traitement le maximum ; on obtient $k = 18$, $\alpha = 0.162$ et une erreur de 0.073%.

Si nos résultats sont probants, il ne s'agit toutefois que d'un cas d'application. Nous n'avons considéré que deux états : présence et absence, sans prendre en compte divers facteurs. Ainsi, nous aurions pu considérer les niveaux d'activité, et étudier s'il y a une différence selon si la personne présente bouge beaucoup ou si elle est presque immobile ou si elle est absente.

Nous n'avons pas non plus eu l'occasion de jouer sur tous les paramètres inhérents à nos expériences : nombre et emplacement des capteurs, durée minimale du jeu de test pour qu'il soit pertinent, quantité d'activité idéale pour le paramétrage optimal... Nous n'avons employé qu'un seul type de capteur (capteur de mouvement), d'une seule marque. En comparant deux capteurs d'une même marque (Figure VI), les résultats ne sont pas identiques, ce qui nous conforte dans l'optique de réaliser les réglages chez l'utilisateur avec ses propres capteurs, et non pas fournir un réglage usine uniquement, ce qui montre bien la nécessité d'une calibration automatique.



(a) Premier capteur (b) Second capteur

FIGURE 2: Comparaison des cartes de températures (k, α) de deux capteurs de même modèle : les bons couples (k, α) ne sont pas les mêmes.

Enfin, nous n'avons appliqué qu'un seul algorithme de classification, k-NN. Si nous supposons qu'un algorithme supervisé est un bon choix, tant par sa mise en œuvre plus aisée en situation réelle que par les résultats que k-NN nous a permis d'obtenir, il pourrait être judicieux de les comparer avec ceux d'autres algorithmes de classification, supervisés ou non.

VII. Conclusion

Le présent rapport démontre que la calibration automatique est non seulement envisageable pour un système domotique basé sur Raspberry Pi, mais aussi recommandée, du fait des disparités entre deux capteurs de même modèle. Elle nous a permis de reconnaître des attributs de contexte simples (absence/présence). Elle représente un gain temporel et financier comparé à la calibration manuelle nécessitant un technicien, tout en offrant de bons résultats (taux d'erreurs inférieurs à 0.01%). Nous avons mis en œuvre deux méthodes pour obtenir un paramétrage optimal pour notre algorithme de classification. Pour un usage domotique, la descente du gradient est la plus optimale, notamment en l'appliquant sur le maximum à trois points de nos données.

Néanmoins, ce fait n'est pas généralisable à tous les capteurs et tous les attributs de contexte. Le maximum sur trois points est optimal dans notre cas d'étude, mais pour d'autre capteurs, il faut envisager à nouveau toutes les variables possibles. De même, les variables à deux dimensions sont plus lentes pour aucun gain notable de qualité dans notre étude, mais peuvent être utiles dans d'autre cas.

Il serait intéressant, d'une part, d'appliquer notre méthode sur une plus grande échelle, c'est-à-dire via des attributs de contexte à plusieurs états, comme par exemple le taux d'activité. D'autre part, nous pourrions travailler à l'amélioration de notre méthode, en testant d'autres types de capteurs ou d'autres algorithmes de classification.

Références

[1] Dey, Anind K. (2001). *Understanding and Using Context*. Personal Ubiquitous Computing.

[2] Bastien Pietropaoli. *Reconnaissance de contexte stable pour l'habitat intelligent*. Ubiquitous Computing. Universite Rennes 1, 2013.

[3] Lloyd., S. P. (1982). *Least squares quantization in PCM*. IEEE Transactions on Information Theory 28 (2) : 129–137.

[4] Sebastien Regis, Andrei Doncescu, Makoto Takizawa and Guillaume Cleuziou. *Initialization of masses by Okm for the belief function theory. Application to System Biology*.

[5] Thierry Denoeux. *A k-nearest Neighbor Classification Rule Based on Dempster-Shafer Theory*

[6] Brian S. Everitt and Torsten Hothorn. "A Handbook of Statistical Analyses Using R". Second Edition, CRC Press, 2010.

[7] Dempster. *Upper and lower probabilities induced by a multivalued mapping*. The Annals of Mathematical Statistics, 38(2), 1967.

[8] Anne-Laure Jousselme, Dominique Grenier and Eloi Bossé. *A new distance between two bodies of evidence*. Information Fusion.

Traitements	Normal	Dérivation	Moyenne	Maximum	Minimum
Normal	$k = 20$ $\alpha = 0.15$ $erreur = 0.128$	$k = 5$ $\alpha = 0.50$ $erreur = 0.108$	$k = 3$ $\alpha = 0.95$ $erreur = 0.066$	$k = 3$ $\alpha = 0.85$ $erreur = 0.069$	$k = 8$ $\alpha = 0.40$ $erreur = 0.115$
Dérivation	$k = 5$ $\alpha = 0.50$ $erreur = 0.108$	$k = 9$ $\alpha = 0.25$ $erreur = 0.122$	$k = 7$ $\alpha = 0.40$ $erreur = 0.086$	$k = 11$ $\alpha = 0.30$ $erreur = 0.083$	$k = 7$ $\alpha = 0.4$ $erreur = 0.102$
Moyenne	$k = 3$ $\alpha = 0.95$ $erreur = 0.066$	$k = 7$ $\alpha = 0.40$ $erreur = 0.086$	$k = 5$ $\alpha = 0.60$ $erreur = 0.063$	$k = 6$ $\alpha = 0.55$ $erreur = 0.069$	$k = 8$ $\alpha = 0.50$ $erreur = 0.078$
Maximum	$k = 3$ $\alpha = 0.85$ $erreur = 0.069$	$k = 11$ $\alpha = 0.30$ $erreur = 0.083$	$k = 6$ $\alpha = 0.55$ $erreur = 0.069$	$k = 9$ $\alpha = 0.40$ $erreur = 0.063$	$k = 5$ $\alpha = 0.75$ $erreur = 0.060$
Minimum	$k = 8$ $\alpha = 0.40$ $erreur = 0.115$	$k = 7$ $\alpha = 0.4$ $erreur = 0.102$	$k = 8$ $\alpha = 0.50$ $erreur = 0.078$	$k = 5$ $\alpha = 0.75$ $erreur = 0.060$	$k = 10$ $\alpha = 0.2$ $erreur = 0.171$

TABLE II: Résultats obtenus pour chaque traitement avec la méthode exhaustive.

Traitements	Normal	Dérivation	Moyenne	Maximum	Minimum
Normal	$k = 18$ $\alpha = 0.134$ $erreur = 0.129$ $temps = 7.4$	$k = 16$ $\alpha = 0.162$ $erreur = 0.120$ $temps = 10.8$	$k = 20$ $\alpha = 0.171$ $erreur = 0.095$ $temps = 13.4$	$k = 17$ $\alpha = 0.167$ $erreur = 0.085$ $temps = 11.1$	$k = 18$ $\alpha = 0.160$ $erreur = 0.127$ $temps = 12.0$
Dérivation	$k = 16$ $\alpha = 0.162$ $erreur = 0.120$ $temps = 10.8$	$k = 15$ $\alpha = 0.147$ $erreur = 0.125$ $temps = 8.9$	$k = 20$ $\alpha = 0.168$ $erreur = 0.092$ $temps = 13.0$	$k = 19$ $\alpha = 0.169$ $erreur = 0.090$ $temps = 12.3$	$k = 19$ $\alpha = 0.163$ $erreur = 0.108$ $temps = 12.4$
Moyenne	$k = 20$ $\alpha = 0.171$ $erreur = 0.095$ $temps = 13.4$	$k = 20$ $\alpha = 0.168$ $erreur = 0.092$ $temps = 13.0$	$k = 14$ $\alpha = 0.157$ $erreur = 0.084$ $temps = 7.9$	$k = 16$ $\alpha = 0.164$ $erreur = 0.089$ $temps = 10.4$	$k = 20$ $\alpha = 0.169$ $erreur = 0.093$ $temps = 12.5$
Maximum	$k = 17$ $\alpha = 0.167$ $erreur = 0.085$ $temps = 11.1$	$k = 19$ $\alpha = 0.169$ $erreur = 0.090$ $temps = 12.3$	$k = 16$ $\alpha = 0.164$ $erreur = 0.089$ $temps = 10.4$	$\boldsymbol{k = 18}$ $\boldsymbol{\alpha = 0.162}$ $\boldsymbol{erreur = 0.073}$ $\boldsymbol{temps = 6.6}$	$k = 20$ $\alpha = 0.172$ $erreur = 0.082$ $temps = 12.4$
Minimum	$k = 18$ $\alpha = 0.160$ $erreur = 0.127$ $temps = 12.0$	$k = 19$ $\alpha = 0.163$ $erreur = 0.108$ $temps = 12.4$	$k = 20$ $\alpha = 0.169$ $erreur = 0.093$ $temps = 12.5$	$k = 20$ $\alpha = 0.172$ $erreur = 0.082$ $temps = 12.4$	$k = 15$ $\alpha = 0.134$ $erreur = 0.175$ $temps = 5.6$

TABLE III: Résultats obtenus pour chaque traitement avec la méthode du gradient.