Localisation Multi-capteurs Tolérante aux Fautes : Validation expérimentale sur un cas réel

Kaci Bader, Walter Schön, Benjamin Lussier

Localisation Multi-capteurs Tolérante aux Fautes : Validation expérimentale sur un cas réel

BADER Kaci, LUSSIER Benjamin, SCHÖN Walter
Université de Technologie de Compiègne
UMR CNRS 7253, Heudiasyc BP 20529
Email : {kaci.bader, benjamin.lussier, walter.schon}@hds.utc.fr

***Résumé*—Dans ce papier, nous présentons une architecture de localisation pour les robots mobiles. Cette architecture est tolérante aux fautes matérielles dans les capteurs de perception et logicielles dans les mécanismes de fusion de données. Elle implémente deux systèmes de localisation indépendants utilisant deux filtres de Kalman diversifiés, et assure la détection d'erreur par la technique de comparaison et le rétablissement du système par la méthode de compensation. Un environnement d'évaluation de l'architecture proposée est mis en place, en utilisant l'acquisition de données réelles avec la plateforme de laboratoire Heudiasyc PACPUS [1], puis le rejeu de ces données avec injection de fautes sous Matlab.**

I. Introduction

La capacité pour le robot de se localiser, c'est-à-dire de connaitre à tout instant sa position et la précision de celle-ci, est essentielle pour de nombreuses fonctionnalités comme la navigation et la planification. Les travaux scientifiques cherchant à résoudre ce problème emploient en général la fusion de données par filtrage de Kalman [7], [10] et [11]. Comme le filtre de Kalman ne peut être appliqué qu'à des systèmes linéaires, le filtre de Kalman étendu a été proposé pour les systèmes non linéaires, et a été mis en œuvre avec succès pour l'estimation de position de robots mobiles dans [5], [10], et [12]. Ces approches de fusion de données emploient plusieurs capteurs de même type afin d'exploiter leur redondance ou des capteurs de types différents afin d'exploiter leurs complémentarité pour réduire les incertitudes et les imprécisions dans les données extraites et améliorer la localisation.

Comme les systèmes robotiques décident leurs actions selon leur perception, assurer la sûreté de fonctionnement de cette perception est critique. Cependant étant donné que les systèmes robotiques évoluent généralement dans des environnements ouverts, les tests présentent des contextes d'exécution quasi-infinis. Ainsi la validation de leur perception devient une tâche difficile et coûteuse. Pour remédier à ces problèmes, des approches de tolérance aux fautes ont été proposées dans la littérature. Ainsi, plutôt que de garantir par des tests exhaustifs que le système de perception ne comporte aucune faute, elles essayent d'assurer que la sortie de la perception est correcte en acceptant la présence de faute dans le système.

Ces dernières approches utilisent principalement la tolérance aux fautes par duplication / comparaison : elles utilisent un modèle analytique du système comme diversification des capteurs ciblés, puis le filtre de Kalman fusionne l'état estimé par ce modèle et les mesures données par les capteurs. Le résidu issu de la fusion de données est utilisé comme indicateur d'erreur. Dans [9], une architecture de filtre de Kalman tolérante aux fautes en perception multi-capteurs a été proposée. Elle détecte les fautes transitoires dans les capteurs en utilisant la technique de résidus, et les fautes permanentes par une méthode de vote [6]. Dans [13] les auteurs incorporent des capteurs abstraits dans un capteur virtuel pour améliorer les caractéristiques des capteurs physiques en présence de bruit et de défaillances. En utilisant un modèle mathématique pour évaluer les données de ces capteurs redondants, cette approche permet d'obtenir une estimation fiable de position des robots mobiles tolérante ainsi les erreurs des capteurs. [1] présente les approches de fusion de données pour les systèmes de réseau inertiels. Dans ces systèmes, les auteurs utilisent des nœuds redondants structurés hiérarchiquement. La plupart des nœuds jouent le rôle d'esclaves tandis que l'un d'entre eux joue le rôle de maître, réalisant la fusion globale par filtrage de Kalman. Les auteurs utilisent la redondance des IMU(Inertial Measurement Unit) pour assurer le masquage de fautes et avoir des sorties plus précises.

À notre avis une lacune importante de ces différentes techniques est qu'elles se concentrent uniquement sur les fautes matérielles dans les capteurs, en s'appuyant sur les composants de filtre de Kalman pour détecter et tolérer les erreurs. Or, en raison de leur programmation déclarative, le comportement de ces algorithmes de filtrage est difficile à prévoir, ce qui complique leur validation par des approches formelles, comme la vérification par la preuve. Par conséquent, dans cet article, nous proposons un mécanisme permettant d'assurer non seulement la tolérance aux fautes matérielles dans les capteurs comme les approches exposées précédemment, mais aussi la tolérance aux fautes logicielles dans les algorithmes de fusion. Ce travail est une continuité du travail présenté dans [3] en implémentant un mécanisme de duplication-comparaison pour une application réelle de localisation des véhicules intelligents. Les contributions primaires de ce travail sont :

- détection et recouvrement d'une faute matérielle : l'architecture proposée détecte et recouvre une erreur matérielle dans les capteurs de perception sans exploiter les mécanismes de fusion.
- détection d'une faute logicielle : elle détecte une erreur

1. *PACPUS : Perception et Assistance pour une Conduite Plus Sure.*

logicielle dans les algorithmes de filtre de Kalman.
- validation expérimentale : résultats expérimentaux sur des données réelles validant notre architecture et sa réaction aux fautes matérielles et logicielles activées.

La suite de cet article est organisée comme suit : après cette introduction présentant le contexte et la motivation de ce travail, la section II présente notre approche de tolérance aux fautes en fusion de données, détaillant les services de détection de faute et de rétablissement du système sur une application de localisation des robots mobiles. La section III présente une évaluation des performances de notre architecture, détaillant l'environnement matériel et logiciel de validation, et montrant la réaction de notre système face à des injections de fautes. Enfin, la section IV conclut l'article, en présentant quelques perspectives de recherche pour les travaux futurs.

II. APPROCHE PROPOSÉE

Dans cette section, nous présentons notre architecture de tolérance aux fautes en fusion de données par filtre de Kalman. Notre approche est basée sur la méthode classique de duplication / comparaison mettant en œuvre deux systèmes de localisation (sous-section II-A1 et II-A2), comme indiqué dans la figure 1. Une présentation détaillée de cette architecture peut être trouvée dans [4]. Ces deux systèmes fonctionnent en parallèle, utilisent des capteurs indépendants et redondants pour percevoir l'état du véhicule et de son environnement, et implémentent deux filtres de Kalman diversifiés pour combiner les sorties de ces capteurs. La configuration présentée sur la figure 1 peut tolérer une erreur matérielle dans l'un des capteurs, et détecter une erreur logicielle dans l'un des filtres de Kalman. Mais une telle erreur logicielle peut être tolérée par la technique de diversification fonctionnelle [2] en diversifiant une troisième version de filtre de Kalman et en utilisant la technique de vote majoritaire. Dans la suite de cette section, nous présentons d'abord les deux modèles de processus et d'observation des deux filtres de Kalman mis en place pour assurer la localisation, et puis nous détaillons les services de tolérance aux fautes fournis.

A. Localisation par filtre de Kalman

l'algorithme de filtre de Kalman implémenté dans les deux systèmes de localisation montrés dans la figure 1 consiste en deux étapes successives, la prédiction et la correction. La phase de prédiction utilise un modèle de processus prenant en entrée la commande exécutée par le robot pour produire une estimation de l'état courant. Dans l'étape de correction, les mesures de l'instant courant sont liées à l'état du système dans un modèle d'observation. Cette observation corrige l'état prédit dans le but d'obtenir une estimation plus précise de la position du robot.

Les équations utilisées dans nos filtres de Kalman sont issues du modèle de robot montré dans la figure 2. Ce robot est constitué de deux roues arrières fixes sur le même axe et une roue avant centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et la direction de la roue orientable, avec : (x, y) : la position, θ : l'angle de lacet, c'est à dire l'angle que fait le corps du véhicule avec l'axe x horizontal, ϕ : l'angle de direction par rapport au corps du véhicule c'est-à-dire l'angle de braquage de la roue. L : la distance entre les essieux du véhicule (la distance entre R et F dans la figure), et e : la distance entre les deux roues arrière.

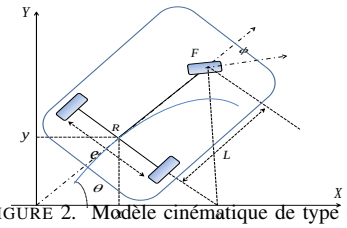

FIGURE 2. Modèle cinématique de type tricycle

1) Localisation par couplage d'un système de navigation inertiel (INS) et d'un système d'odométrie avant (F-odo): Le premier système de localisation mis en œuvre consiste à coupler un système inertiel et un système d'odométrie de la roue avant du véhicule comme montré dans la figure 1. Nous utilisons les données d'un système inertiel (INS) pour prédire l'état du système. Cet état sera mis à jour et corrigé par les données du système d'odométrie de la roue avant noté *F-odo*. Ce système odométrique estime la position en employant le modèle cinématique tricycle ; il utilise la vitesse de la roue avant du véhicule V_F fournie par le capteur odométrique avant du véhicule (F-WSS) et l'angle de braquage ϕ délivré par le capteur dédié. Les variables d'état sont les positions du véhicule x, y, son orientation θ et sa vitesse longitudinale V. Les modèles de processus et d'observation de ce premier filtre sont les suivants :

- *Modèle du processus* : nous utilisons l'équation (1) comme un modèle de système dans KF_1, prenant en entrée $U_{INS}(\gamma_{INS}, \omega_{INS})$, l'accélération longitudinale γ_{INS} mesurée par l'accéléromètre et la vitesse angulaire ω_{INS} mesurée par le gyromètre.

$$\underbrace{\begin{bmatrix} x_{KF_1} \\ y_{KF_1} \\ \theta_{KF_1} \\ V_{KF_1} \end{bmatrix}_k}_{X_{KF_{1_k}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & dt_k cos(\theta_{KF_1}) \\ 0 & 1 & 0 & dt_k sin(\theta_{KF_1}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{k-1}}_{A} \cdot \underbrace{\begin{bmatrix} x_{KF_1} \\ y_{KF_1} \\ \theta_{KF_1} \\ V_{KF_1} \end{bmatrix}_{k-1}}_{X_{KF_{1_{k-1}}}} + \underbrace{\begin{bmatrix} \frac{1}{2}dt_k^2 & 0 \\ \frac{1}{2}dt_k^2 & 0 \\ 0 & 1 \\ dt_k & 0 \end{bmatrix}_{k-1}}_{B} \cdot \underbrace{\begin{bmatrix} \gamma_{INS} \\ \omega_{INS} \end{bmatrix}_{k-1}}_{U_{INS_{k-1}}} + w_{KF_{1_k}} \quad (1)$$

  $w_{KF_{1_k}}$ est le bruit du système dont la matrice de covariance est Q_{KF_1}.
- *Modèle d'observation* : ce modèle est basé sur le capteur odométrique de la roue avant du véhicule et un modèle de type tricycle [8]. Les sorties odométriques F-odo sont liées à l'état du système selon l'équation 2.

$$Z_{KF_{1_k}} = \begin{bmatrix} x_{Fodo} \\ y_{Fodo} \\ \theta_{Fodo} \\ V_F \end{bmatrix}_k = H_{KF_1} . X_{KF_{1_k}} + v_{KF_{1_k}} \quad (2)$$

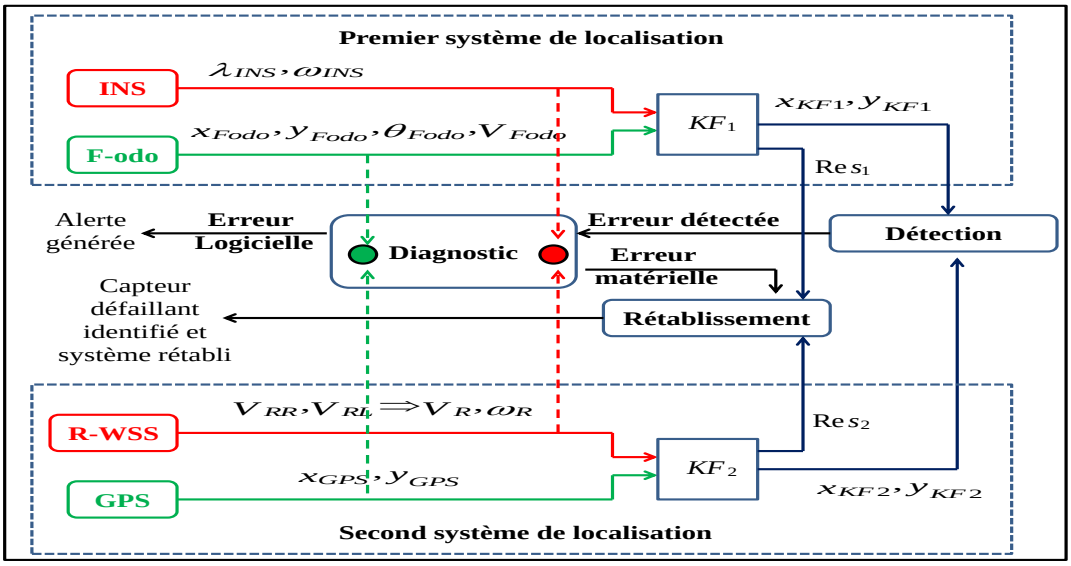

FIGURE 1. Architecture de tolérance aux fautes pour une localisation des robots mobiles

$(x_{Fodo}, y_{Fodo}, \theta_{Fodo})$ est la pose du véhicule estimée par le système d'odométrie avant, et V_F est la vitesse longitudinale fournie par l'encodeur avant (F-WSS). La matrice d'observation H_{KF_1} utilisée dans le modèle d'observation est la matrice d'identité puisque les mesures sont les mêmes que les variables d'état du système. $v_{KF_{1_k}}$ est le bruit de mesures dont la matrice de covariance est R_{KF_1}.

2) Localisation par couplage des capteurs odométriques arrière R-WSS et d'un GPS: Le second système de localisation mis en œuvre dans cet article consiste à coupler les capteurs odométriques des roues arrière du véhicule (R-WSS) avec un capteur GPS par un filtre de Kalman. Les détails d'implémentation de ce filtre sont décrits ci-dessous :

- *Modèle du processus* : le modèle du système est décrit dans les équations (3) :

$$\underbrace{\begin{bmatrix} x_{KF_{2_k}} \\ y_{KF_{2_k}} \\ \theta_{KF_{2_k}} \\ V_{KF_{2_k}} \end{bmatrix}}_{X_{KF_{2_k}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} x_{KF_{2_{k-1}}} \\ y_{KF_{2_{k-1}}} \\ \theta_{KF_{2_{k-1}}} \\ V_{KF_{2_{k-1}}} \end{bmatrix}}_{X_{KF2_{k-1}}} + \underbrace{\begin{bmatrix} dt_k cos(\theta_k) & 0 \\ dt_k sin(\theta_k) & 0 \\ 0 & dt_k \\ 1 & 0 \end{bmatrix}}_{B} \cdot \underbrace{\begin{bmatrix} V_{R_{k-1}} \\ \omega_{R_{k-1}} \end{bmatrix}}_{U_{R_{k-1}}} + w_{KF_{2_k}} \quad (3)$$

V_R et ω_R sont respectivement la vitesse linéaire et vitesse angulaire du véhicule. Ces valeurs sont calculées à partir des sorties des encodeurs des roues arrière.

◇ V_R : est la moyenne des vitesses droite et gauche des roues arrière.

$$V_R = \frac{V_{RR} + V_{RL}}{2} \quad (4)$$

◇ ω_R est la rotation élémentaire des deux roues arrière

$$\omega_R = \frac{V_{RR} - V_{RL}}{e} \quad (5)$$

où V_{RR} et V_{RL} sont respectivement la vitesse de la roue arrière droite et la vitesse de la roue arrière gauche. $w_{KF_{2_k}}$ est le bruit du système dont la matrice de covariance est Q_{KF_2} :

- *Modèle d'observation* : Les mesures GPS (x_{GPS}, y_{GPS}) sont liées à l'état du système par le modèle d'observation 6.

$$Z_{KF_{2_k}} = \begin{bmatrix} x_{GPS} \\ y_{GPS} \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_{KF_2}} . X_{KF2_k} + v_{KF_{2_k}} \quad (6)$$

Où $v_{KF_{2_k}}$ est le bruit de mesure GPS dont la matrice de covariance est R_{KF_2}.

B. Services de tolérance aux fautes

Dans cette section nous présentons les services de détection de fautes et de rétablissement du système assurés par notre architecture.

1) Détection d'erreur : Pour détecter d'éventuelles erreurs dans le système, nous comparons les positions estimées par les deux filtres de Kalman implémentés. Pour cela nous calculons la distance euclidienne $dis_{(KF_1,KF_2)}$ (équation 7) entre la position (x_{KF_1}, y_{KF_1}) sortie du premier filtre et la position (x_{KF_2}, y_{KF_2}) sortie du second filtre, et comparons ensuite cette distance à un seuil de détection spécifique $Thrs_{Det}$. Si la distance $dis_{(KF_1,KF_2)}$ est supérieure au seuil $Thrs_{Det}$, cela implique qu'un écart significatif entre les deux positions $((x_{KF_1}, y_{KF_1})$ et $x_{KF_2}, y_{KF_2}))$ est survenu. Nous identifions cet écart à la présence d'une erreur. Dans le cas contraire aucune erreur n'est détectée.

$$dis_{(KF_1,KF_2)} = \sqrt{(x_{KF_1} - x_{KF_2})^2 + (y_{KF_1} - y_{KF_2})^2} \quad (7)$$

2) Diagnostic et rétablissement du système: Pour diagnostiquer l'erreur détectée précédemment, et déduire, en cas de faute matérielle, la sortie correcte du système, deux étapes sont nécessaires :

- Comparaison des sorties des capteurs : elle identifie, en cas de faute matérielle, le type de capteur erroné en comparant les sorties des capteurs similaires. C'est à dire les capteurs utilisés dans les modèles de processus en équations 1 et 3 (INS, R-WSS) d'un coté, et les capteurs utilisés dans les modèles d'observation en équation 2 et 6 (F-odo, GPS) de l'autre coté.
  - ◇ Si la sortie d'un capteur s'écarte significativement de son dual, le système diagnostique une faute matérielle sur l'un de ces deux capteurs de même type. Le capteur défectueux sera alors identifié dans la comparaison des résidus.
  - ◇ Si les sorties des capteurs sont similaires, le système diagnostique une faute logicielle. En l'absence de plus de redondance, il n'est pas possible de pousser plus loin le diagnostic (identifier le filtre fautif) mais seulement de mettre le système erroné dans un état sûr. Cependant un autre filtre de Kalman diversifié KF_3 pourrait être mis en œuvre pour diagnostiquer le filtre défectueux, et rétablir le système.
- Comparaison des résidus : elle identifie dans le cas d'une faute matérielle la branche erronée d'après la valeur des résidus des filtres de Kalman. En effet, ce résidu correspond au degré de consistance entre les données fusionnées, une valeur haute indiquant que les données des capteurs fusionnés sont en conflit. Nous nous appuyons ici sur le mécanisme de fusion de données, car nous avons déjà vérifié dans l'étape « Comparaison des sorties des capteurs » que l'écart constaté entre les deux branches est dû à une défaillance de capteurs et non des mécanismes de fusion.

Connaissant maintenant le type de capteur défectueux et la branche erronée correspondante, le système a identifié le capteur erroné, et peut ainsi être rétabli à l'aide des valeurs de position de la branche sans erreur.

III. Évaluation des performance

Dans cette section, nous proposons une étude cherchant à évaluer l'implémentation décrite dans la section II en termes de sûreté de fonctionnement, et à montrer l'efficacité de ces mécanismes de tolérance aux fautes confrontés à des fautes. Nous présentons tout d'abord l'environnement d'expérimentation, et les principes que nous avons utilisés pour réaliser les expériences nécessaires à cette évaluation. Puis nous présentons les résultats de notre campagne d'injection de fautes.

A. Environnement Matériel : Véhicule Carmen et ses capteurs

Le véhicule utilisé pour nos expérimentation est le véhicule Carmen (Citroën C5 Break montré dans la Figure 3) du laboratoire Heudiasyc. Ce véhicule est instrumenté pour la perception et est équipé de plusieurs capteurs : les capteurs odométriques R-WSS, F-WSS, un système de positionnement IMU-GPS Novatel SPAN CPT, et un récepteur GPS haute qualité Septentrio PolarX, et d'autres. Tous ces capteurs sont connectés à un PC via différentes interfaces (Série, Ethernet, USB et Firewire) pour pouvoir faire une sauvegarde des données acquises.

FIGURE 3. Véhicule d'expérimentation (CARMEN)

B. Environnement logiciel : Plate forme PACPUS

Notre environnement logiciel repose sur l'acquisition de données réelles au moyen de la plate forme logicielle PACPUS. Le rejeu de ces données ainsi que l'injection de fautes sont réalisées sous Matlab.

Nous avons effectués l'acquisition de donnée par le véhicule de laboratoire Carmen. Ce dernier parcourt le circuit de test montré sur la Figure 4. Ce circuit d'expérimentation est composé d'une ligne droite suivie d'un rond point. Nous avons choisi ce circuit car il donne un bon exemple d'application pour montrer les différentes situations auxquelles peut être confronté un véhicule dans son environnement, et simule les différentes manœuvres qui peuvent êtres demandées au robot, à savoir le parcourt d'une ligne droite, et des virages sur les ronds points.

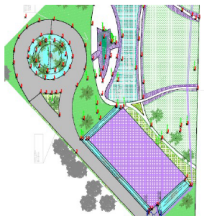

FIGURE 4. Circuit d'expérimentation

Après expérimentation sur le terrain, nous avons rejoués les données acquises en les exportant dans des matrices Matlab, et nous avons réalisé notre campagne d'injection de fautes.

C. Relevés et Mesures

Pour l'application décrite dans cet article nous avons extrait plusieurs relevés, à partir desquels, nous avons défini de nombreuses mesures significatives représentant l'efficacité de l'architecture proposée.

- *Le délai de détection* (Del_{Det}) est la différence entre l'instant d'injection de la faute t_{Inj} et l'instant de détection t_{Det} (i.e. l'instant ou la distance dis_{KF_1,KF_2} dépasse le seuil de détection prédéfini $Thrs_{Det}$).

$$\begin{cases} Del_{Det} = t_{Det} - t_{Inj} \\ \text{Avec} \\ t_{Det} = \underset{dis_{KF_1,KF_2} \geq Thrs_{Det}}{t_k} \end{cases} \quad (8)$$

Cette mesure nous permettra d'avoir un avis sur le délai de détection, car la faute injectée doit vite être détectée pour qu'elle soit diagnostiquée et recouverte avant qu'une seconde faute soit survenue.

Pour déterminer si la faute injectée est bien détectée, diagnostiquée et identifiée, et que le système est correctement rétabli, nous avons défini trois variables booléennes pour chacune des expériences effectuées :

- B_d caractérise si la faute est détectée ($B_d = 1$ si la détection a lieu, $B_d = 0$ sinon),
- B_i représente le fait qu'une faute est bien identifiée et que le composant fautif est bien diagnostiqué ($B_i = 1$ si le diagnostic est correct, $B_i = 0$ sinon),
- B_r détermine si le rétablissement du système après détection est réalisé ($B_r = 1$ si le système est correctement rétabli, $B_r = 0$ sinon).

Nous avons aussi défini certains pourcentages caractérisant notre validation en termes d'identification et de rétablissement du système.

- P_i est le pourcentage de fautes détectées correctement diagnostiquées et identifiées (au sens du diagnostic du paragraphe II-B2).

$$P_i = \frac{\text{Nombre de fautes correctement identifiées}}{\text{Nombre de fautes detectées}} \quad (9)$$

- P_r : est le pourcentage pour lequel le système est bien rétabli (au sens du paragraphe II-B2).

$$P_r = \frac{\text{Nombre de fois que le systeme est correctement rétabli}}{\text{Nombre de fautes detectées}} \quad (10)$$

D. Résultats d'expérimentation

Nous présentons dans ce qui suit un échantillon de notre campagne d'expérimentation. En effet dans cette dernière nous avons réalisé 110 injections de fautes reparties en 90 fautes matérielles et 20 mutations logicielles. Nous exposons d'abord le comportement nominal du système, et ensuite les résultats de trois expériences en présence de trois fautes : la première et la seconde sont des fautes matérielles injectées dans la sortie du GPS et de l'encodeur avant du véhicule F-WSSS, et la troisième est une faute logicielle injectée dans le filtre de Kalman KF_2.

1) Comportement nominal sans faute injectée: Une expérience sans faute a été réalisée pour caractériser le comportement nominal du système. Ces résultats servent comme échantillon de référence pour la comparaison au comportement en présence de fautes. Ils nous sont utiles pour fixer les différents seuils de détection et de diagnostic employés dans notre architecture, ces seuils sont déterminés empiriquement et ont été choisis pour être significativement plus élevés que les valeurs observées dans le comportement nominal.

La Figure 5 représente la position du véhicule estimée par les deux filtres de Kalman (x_{KF_1}, y_{KF_1}), (x_{KF_2}, y_{KF_2}) et la sortie de notre architecture (x_S, y_S), qui est la moyenne de ces deux estimations. La Figure 6 représente la distance $dis_{(KF_1,KF_2)}$. Dans ce cas nominal cette distance est inférieure au seuil de détection choisi $Thrs_{Det}$ (10 m).

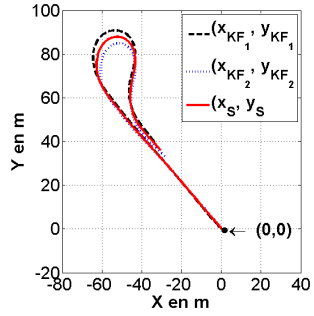

FIGURE 5. Comportement nominal : La position estimée par les deux filtres de Kalman et le système

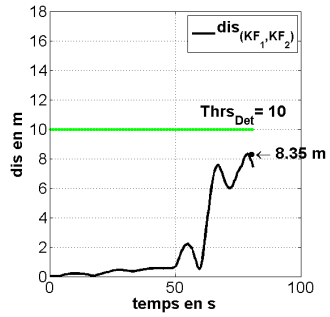

FIGURE 6. Comportement nominal : Distance entre les positions des deux filtres de Kalman

2) Comportement en présence de fautes: Afin d'évaluer la performance et l'efficacité des mécanismes de détection et de rétablissement proposés, nous injectons des fautes matérielles par altération des sorties des capteurs de perception, et des fautes logicielles par mutation directement dans les modèles de processus et d'observation des filtres de Kalman. Elles sont introduites volontairement dans le code pour perturber le fonctionnement de l'architecture à des instants bien précis. Nous présentons ici les résultats de trois injections de fautes : deux matérielles dans le GPS et le capteur odométrique F-WSS et l'autre logicielle dans le filtre de Kalman :

- **Faute additive sur le GPS** : La première faute injectée que nous considérons est une faute additive permanente dans la sortie du GPS. Cette faute externe typique dans la réalité simule un saut dans la position fournie par le GPS en raison de rebonds d'un des signaux satellites. Cette faute n'est généralement pas permanente, mais elle peut se produire pendant un temps significatif. A l'instant $t_i = 60.5$ secondes, nous avons ajouté un saut de 10 mètre sur la composante x_{GPS}, comme décrit dans l'équation (11).

$$x_{GPS}(t_k) = \underset{t_i \leq t_k}{x_{GPS}(t_i)} + 10 \quad (11)$$

Dans la Figure 7, nous voyons que la différence entre les positions estimées par les deux filtres de Kalman

$dis_{(KF_1,KF_2)}$ dépasse le seuil $Thrs_{Det}$, donc le système détecte la présence d'une erreur à l'instant $t_{Det} = 62.5$ secondes.

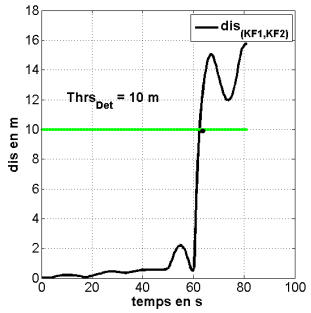

FIGURE 7. Faute de saut sur le GPS : Distance entre les positions des deux filtres de Kalman

Nous voyons dans la figure 8 que le système INS et les encodeurs des roues arrières R-WSS ont des sorties similaires, tandis que la différence entre la position fournie par le GPS et celle délivrée par le système odométrique avant F-odo dépasse le seuil $Thrs_{Pos_{GPS,F-odo}}$. Cela indique que l'erreur détectée dans le système est une erreur matérielle liée soit au GPS, soit aux encodeurs avant du véhicule.

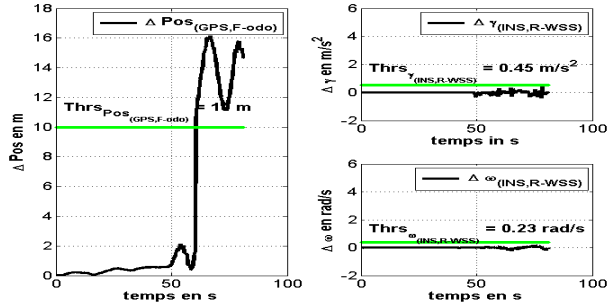

FIGURE 8. Faute de saut sur le GPS : Comparaison des sorties des capteurs

La Figure 9 confirme que la valeur absolue du résidu issu de KF_2 ($\mid Res_{KF_2} \mid$) est nettement plus élevée que la valeur absolue du résidu issu de KF_1 ($\mid Res_{KF_1} \mid$), indiquant que la branche KF_2 contient une erreur.
Connaissant maintenant le couple (GPS, F-odo) contenant le capteur défectueux et la branche erronée, le système peut identifier que le capteur erroné est le GPS, et prendre la sortie correcte (x_{KF_1}, y_{KF_1}) à t = 62,5 secondes, comme indiqué dans la figure 10, soit 2 secondes après l'injection de la faute.

- **Faute de biais sur les encodeurs F-WSS**
  La seconde faute injectée est une faute de biais sur les encodeurs, en ajoutant un biais de 1.5 mètre/seconde sur la sortie du capteur odométrique F-WSS.
  La figure 11 montre que la faute injectée est correctement détectée. La figure 12 indique que la faute détectée

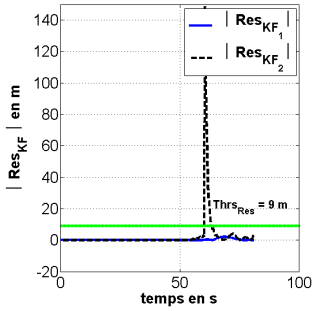

FIGURE 9. Faute de saut sur le GPS : Comparaison des résidus

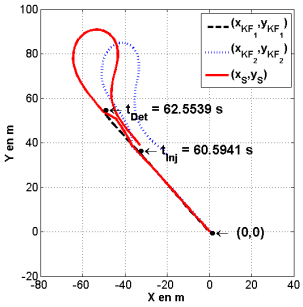

FIGURE 10. Faute de saut sur le GPS : La position estimée par les deux filtres de Kalman et le système

est une faute matérielle liée au capteur GPS ou F-WSS. Cependant les deux résidus $\mid Res_1 \mid$ et $\mid Res_2 \mid$ (Figure 13) ne dépassent pas le seuil $Thrs_{Res}$ prédéfini. Dans ce cas l'erreur évolue trop lentement pour avoir une évolution rapide des résidus et par conséquent la comparaison des résidus ne fait pas ressortir l'erreur. Cela met en évidence une nouvelle limite de l'utilisation des résidus pour la détection de fautes car cela peut conduire comme ici à un diagnostic incomplet. Ainsi nous n'arrivons pas à distinguer lequel des capteurs (GPS ou F-WSS) est fautif, et par conséquent la branche erronée. Pour mettre le système en état sûr, une alerte est générée indiquant que le système de localisation mis en place est défaillant comme montré sur la figure 14.

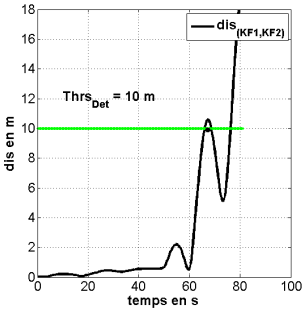

FIGURE 11. Faute de biais sur les encodeurs F-WSS : Distance entre les positions des deux filtres de Kalman

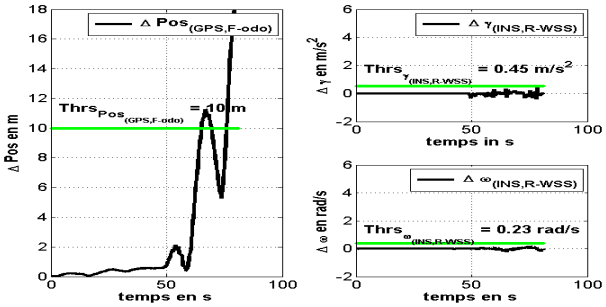

FIGURE 12. Faute de biais sur les encodeurs F-WSS : Comparaison des sorties des capteurs

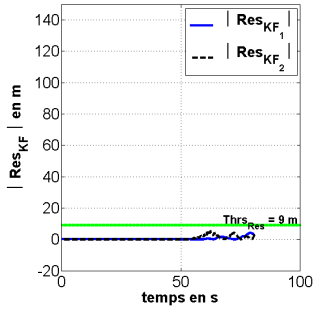

FIGURE 13. Faute de biais sur les encodeurs F-WSS : Comparaison des résidus

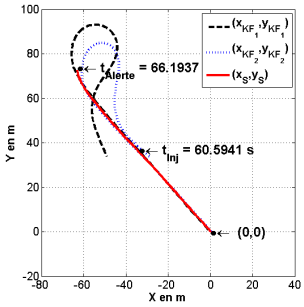

FIGURE 14. Faute de biais sur les encodeurs F-WSS : La position estimée par les deux filtres de Kalman et le système

- **Faute logicielle sur le filtre** KF_2 : Pour les fautes logicielles notre campagne d'injection de fautes utilise la technique de mutation. C'est-à-dire une modification syntaxique élémentaire du code. Cette mutation peut être faite soit sur le modèle de processus ou sur le modèle d'observation des deux filtres de Kalman. Un exemple de mutations réalisée dans notre expérimentation est représenté dans (12). Cette mutation remplace la valeur du premier élément de la matrice d'observation H_{KF_2} initialement égale à 1 par une valeur de 0.8. Cette mutation simule un exemple d'erreur de programmation dans le développement du modèle.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_{KF_2}} \Longrightarrow_{Mutation} \underbrace{\begin{bmatrix} 0.8 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_{KF_2}} \quad (12)$$

Comme on le voit dans la figure 15, la position estimée par KF_1 diverge sensiblement de celle estimée par KF_2 à l'instant t_{Det} = 59,47 secondes. La Figure 16 montre que l'erreur détectée n'est pas causée par une faute matérielle dans les capteurs de perception, indiquant ainsi une faute logicielle dans l'un des algorithmes de filtres de Kalman (KF_1 ou KF_2) sans être en mesure de distinguer lequel. Après la détection d'une telle faute logicielle, notre architecture génère une alerte indiquant que le système de localisation est défaillant comme le montre la figure 17.

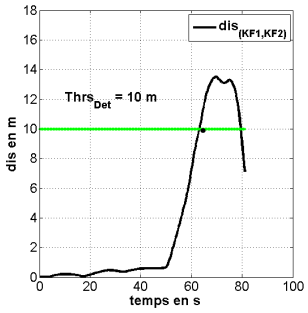

FIGURE 15. Faute logicielle dans KF_2 : Distance entre les positions des deux filtres de Kalman

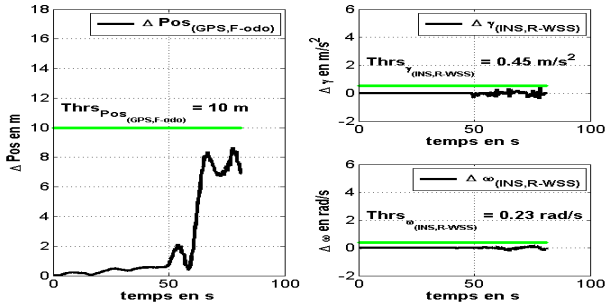

FIGURE 16. Faute logicielle dans KF_2 : Comparaison des capteurs

E. Mesures globales

Dans nos expériences, un grand nombre de fautes sur les capteurs et les algorithmes de filtre de Kalman ont été injectées. Pour chaque type de capteur nous avons injecté un ensemble de fautes de manières différentes, à des instants différents, afin de valider aussi largement que possible leurs conséquences sur le résultat final de l'architecture, et mesurer la capacité de notre approche en terme de détection et de rétablissement. Les tables I, II montrent un échantillon de nos injections de fautes matérielles sur les capteurs GPS et F-WSS respectivement, et la table III montre un échantillon de mutations logicielles réalisées sur les deux filtres de Kalman.

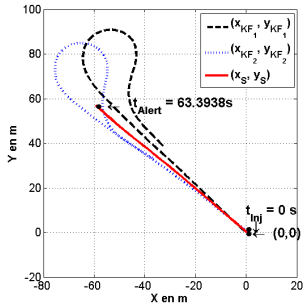

FIGURE 17. Faute logicielle dans KF_2 : La position estimée par les deux filtres de Kalman et le système

Comp	Saut	Dir	t_{Inj}	Del_{Det}	B_d	B_i	B_r
GPS	2	-X	60.59	X	1	0	0
GPS	5	+X	72.35	5.04	1	0	0
GPS	10	-Y	60.69.15	1.40	1	0	0
GPS	20	+Y	72.35	2.24	0	0	0

TABLE I
INJECTION DE FAUTES SUR LE GPS

Comp	Bias	t_{Inj}	Del_{Det}	B_d	B_i	B_r
F-WSS	1	60.59	3.92	1	1	1
F-WSS	1.5	72.35	5.60	1	1	0
F-WSS	2	72.35	5.32	1	1	1

TABLE II
INJECTION DE FAUTES DE BIAIS SUR LES ENCODEURS AVANT F-WSS

Original	Mutant	t_{Inj}	Del_{Det}	B_d	B_i	B_r
$A^1_{1,4} : cos(\theta)$	$A^1_{1,4} : sin(\theta)$	0.00	72.63	1	0	0
$H^1_{1,2} : 0$	$H^1_{1,2} : 1$	0.00	56.11	1	0	0
$A^2_{1,1} : 1$	$A^2_{1,1} : 0$	0.00	55.83	1	0	0
$B^2_{1,1} : dt$	$B^2_{1,1} : dt/2$	0.00	X	0	0	0
$H^2_{1,2} : 0$	$H^2_{1,2} : 1$	0.00	55.55	0	0	0

TABLE III
INJECTION DE FAUTES LOGICIELLES PAR MUTATION

D'après les traces d'exécution et les différents relevés, sur les 90 fautes matérielles injectées sur les capteurs GPS, F-WSS, et l'angle de braquage, toutes les fautes détectées sont diagnostiquées, cependant seulement 62.32 % sont recouvertes et permettent au système d'être rétabli. Cet écart de performance est dû principalement au choix des seuils de rétablissement utilisés. En effet, nous avons choisi empiriquement tous les seuils de façon a ce qu'ils excédent les valeurs de comportement nominal. Pour améliorer les performances de notre architecture une étude approfondie de ces seuils devient indispensable. Sur les 20 mutations que nous avons simulées 90 % des fautes ayant une incidence sur le système ont été détectées par notre architecture. Comme mentionné à plusieurs reprises, étant donné que le niveau de redondance dans notre architecture est limité à deux branches, ces fautes logicielles détectées ne sont pas rétablies. Pour assurer ce service de rétablissement une troisième branche diversifiée est nécessaire, en employant la technique de vote majoritaire.

IV. CONCLUSION

Dans ce papier nous avons présenté une application de localisation tolérante aux fautes pour les robots mobiles. Elle implémente une méthode de duplication / comparaison pour assurer des services de tolérance aux fautes en fusion de données multi-capteurs. Nous avons mis en place un environnement d'évaluation des mécanismes proposés (acquisition de données avec PACPUS, rejeu de données et injection de fautes avec Matlab). Les résultats d'expérimentation montrent l'efficacité de nos mécanismes en termes de détection, du rétablissement de fautes matérielles dans les capteurs de perception, et de fautes logicielles dans les algorithmes de filtre de Kalman implémentés. Dans nos futur travaux nous envisageons l'amélioration de nos mécanismes en :

- injectant un grand nombre de fautes pour évaluer d'avantage notre architecture.
- implémentant un troisième filtre de Kalman diversifié de ceux présentés ici pour rétablir le système d'une faute logicielle.
- ajoutant plus de redondance matérielles et logicielles pour éliminer l'hypothèse de faute unique, et traiter les fautes multiples.

V. REMERCIEMENT

Ce travail a été réalisé et financé dans le cadre du LABEX MS2T et de l'EQUIPEX ROBOTEX. Il a été soutenu par le gouvernement français, à travers les programmes " Investissements d'avenir " gérés par l'Agence Nationale de la Recherche (Références : ANR-11-IDEX-0004-02 et ANR-10-EQPX-44).

RÉFÉRENCES

[1] David J Allerton et al. Distributed data fusion algorithms for inertial network systems. *Radar, Sonar & Navigation, IET*, 2(1) :51–62, 2008.

[2] Algirdas Avizienis et al. On the implementation of n-version programming for software fault tolerance during execution. In *the First IEEE-CS International Computer Software and Applications Conference (COMPSAC 77), Chicago*, 1977.

[3] Kaci Bader et al. Architecture de filtre de kalman tolérante aux fautes pour la localisation des robots mobiles. *Lambda Mu 19*, 2014.

[4] Kaci Bader et al. A fault tolerant architecture for data fusion targeting hardware and software faults. In *The 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2014.

[5] Oscar Laureano Casanova et al. Robot position tracking using kalman filter. In *the World Congress on Engineering,London UK*, volume II, 2008.

[6] P. Caspi et al. Threshold and bounded-delay voting in critical control systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 327–337. Springer, 2000.

[7] Byoung-Suk Choi et al. The position estimation of mobile robot under dynamic environment. In *the 33rd Annual Conference of the IEEE Industrial Electronics Society, IECON*, pages 134–138, 2007.

[8] Alessandro De Luca et al. Feedback control of a nonholonomic car-like robot. In *Robot motion planning and control*, pages 171–253. Springer, 1998.

[9] PJ Escamilla-Ambrosio et al. A hybrid kalman filter-fuzzy logic multisensor data fusion architecture with fault tolerant characteristics. In *the international conference on artificial intelligence*, pages 361–367, 2001.

[10] Leonie Freeston. Applications of the kalman filter algorithm to robot localisation and world modelling. *Electrical Engineering Final Year Project. University of Newcastle, Australia*, 2002.

[11] Congwei Hu et al. Adaptive kalman filtering for vehicle navigation. *Journal of Global Positioning Systems*, 2(1) :42–47, 2003.

[12] Son-Goo Kim et al. Kalman filtering for relative spacecraft attitude and position estimation. *Journal of Guidance, Control, and Dynamics*, 30(1) :133–143, 2007.

[13] S. Zug et al. An approach towards smart fault-tolerant sensors. In *the International Workshop on Robotic and Sensors Environments. ROSE*, pages 35–40, 2009.