RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ
INDUSTRIELLE

Brevet d'invention

Code de la propriété intellectuelle - Livre VI

DÉCISION DE DÉLIVRANCE

Le Directeur général de l'Institut national de la propriété industrielle décide que le brevet d'invention n° **13 55645** dont le texte est ci-annexé est délivré à :
CENTRE NATIONAL D'ETUDES SPATIALES Etablissement public

La délivrance produit ses effets pour une période de vingt ans à compter de la date de dépôt de la demande, sous réserve du paiement des redevances annuelles.

Mention de la délivrance est faite au Bulletin officiel de la propriété industrielle n° 17/07 du 17.02.17 (n° de publication 3 007 232).

Fait à Courbevoie, le 17.02.17

Le Directeur général de l'Institut national
de la propriété industrielle

Romain SOUBEYRAN

Siège : 15 rue des Minimes - CS 50001 - 92677 COURBEVOIE Cedex - Téléphone : **0 820 210 211** Service 0,10 € / min + prix appel - Télécopie : +33 (0)1 56 65 86 00
www.inpi.fr - contact@inpi.fr
Établissement public national créé par la loi n°51 - 444 du 19 avril 1951

RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

COURBEVOIE

(11) N° de publication : **3 007 232**

(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **13 55645**

(51) Int Cl[8] : ***H 04 L 1/00*** *(2017.01)*

BREVET D'INVENTION B1

PROCEDE DE DETECTION ET/OU DE CORRECTION AUTOMATIQUE D'ERREURS DANS UN FLUX DE DONNEES MULTIPLEXEES.

(22) **Date de dépôt :** 17.06.13.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 19.12.14 Bulletin 14/51.

(45) **Date de la mise à disposition du public du brevet d'invention :** 17.02.17 Bulletin 17/07.

(56) **Liste des documents cités dans le rapport de recherche :**

Se reporter à la fin du présent fascicule

(60) **Références à d'autres documents nationaux apparentés :**

(○) **Demande(s) d'extension :**

(71) **Demandeur(s) :** *CENTRE NATIONAL D'ETUDES SPATIALES Etablissement public* — FR.

(72) **Inventeur(s) :** TOURNERET JEAN YVES, WENDT HERWIG et DOBIGEON NICOLAS.

(73) **Titulaire(s) :** CENTRE NATIONAL D'ETUDES SPATIALES Etablissement public.

(74) **Mandataire(s) :** OFFICE ERNEST T. FREYLINGER S.A..

Domaine de l'invention

[0001] La présente invention porte de manière générale sur la détection et/ou la correction automatique d'erreurs dans un flux de données multiplexées, c'est-à-dire une séquence d'échantillons de données appartenant à une pluralité de canaux entrelacés.

Problème technique

[0002] Dans un flux de données contenant des échantillons de données appartenant à des canaux entrelacés de manière cyclique, tout échantillon de données supplémentaire qui peut être introduit dans le flux de données perturbe la périodicité de l'entrelacement.

[0003] Dans la suite, le nombre de canaux entrelacés sera appelé N_c. Une séquence de N_c échantillons de données dans le flux de données sera appelée une trame. Dans l'idéal, chaque canal se voit ainsi attribuer le même intervalle à l'intérieur de chaque trame, faisant du démultiplexage ou du désentrelacement une tâche relativement aisée. Des échantillons de données supplémentaires, quelle que soit leur source, résultent non seulement en valeurs aberrantes (appelées "glitchs") dans le flux de données mais également en permutations cycliques des canaux en aval des échantillons de données supplémentaires, c'est-à-dire une discordance ou un décalage des échantillons de données qui suivent les échantillons de données supplémentaires par rapport aux intervalles dans lesquels ils devraient en fait se trouver.

[0004] Un objet de la présente invention est de fournir un procédé de détection automatique ou détection et de correction (suppression) de tels échantillons de données supplémentaires.

Description générale de l'invention

[0005] Un procédé de détection et/ou de correction automatique d'erreurs dans un flux de données multiplexées est proposé. Le procédé comprend l'obtention (par exemple par l'intermédiaire d'un canal de communication physique ou à partir d'une mémoire) d'un flux de données multiplexées contenant des échantillons de données, le flux de données comprenant une

pluralité de canaux de données entrelacés organisés de manière cyclique en trames, chaque canal de données possédant un intervalle attribué à l'intérieur de chaque trame, dans lequel intervalle se trouve un échantillon de données appartenant vraisemblablement au canal. Les échantillons de données transportés par le flux de données comprennent des échantillons de données ordinaires (chacun d'eux pouvant être attribué à l'un des canaux entrelacés) et des échantillons de données supplémentaires (glitchs qui ne peuvent pas être attribués à l'un quelconque des canaux entrelacés), ce qui provoque un décalage des échantillons de données ordinaires subséquents par rapport aux intervalles. Selon la présente invention, les échantillons de données supplémentaires sont identifiés par l'exécution d'un algorithme de Viterbi. En tant qu'états (dits "cachés"), associés à chaque échantillon de données, l'algorithme de Viterbi utilise les décalages possibles (c'est-à-dire un décalage d'un intervalle, deux intervalles, etc.) provoqués par des échantillons de données supplémentaires reçus jusqu'à l'échantillon de données associé. Des transitions autorisées entre états comprennent une première transition (c'est-à-dire une transition d'un premier type), représentant une réception d'un échantillon de données ordinaire et, par conséquent, une constance de décalage, et une deuxième transition (c'est-à-dire une transition d'un deuxième type), représentant une réception d'un échantillon de données supplémentaire et, par conséquent, un accroissement de décalage d'une unité (un intervalle).

[0006] Comme on pourra l'observer, le procédé de détection et/ou de suppression de glitchs exploite précisément la nature séquentielle du mécanisme de corruption (permutation cyclique des canaux de données à l'intérieur des trames, c'est-à-dire discordance entre les canaux de données et leurs intervalles attribués respectivement du fait de la présence de glitchs) afin de récupérer les échantillons de données ordinaires qui sont effectivement présents dans le flux de données corrompu et restaurer l'ordre de canaux correct. Des modes de réalisation des procédés se sont avérés très efficaces dans des expériences de simulation, dans lesquelles des flux de données initiaux étaient artificiellement corrompus : 99,1% à 99,9% des données corrigées correspondaient précisément aux données initiales non corrompues.

Un avantage intéressant du procédé est qu'il n'exige aucune interpolation ou extrapolation de données et par conséquent qu'il ne modifie pas le contenu informatif du flux de données. Cet aspect peut être particulièrement utile dans la correction de mesures scientifiques.

[0007] De préférence, le procédé comprend une étape dans laquelle un flux de données multiplexées corrigé, qui diffère du flux de données multiplexées en ce qu'il ne contient pas les échantillons de données supplémentaires identifiés, est produit (et, par exemple, affiché et/ou conservé en mémoire dans une mémoire informatique).

[0008] Il est inhérent à l'algorithme de Viterbi que chaque transition autorisée possède un poids (ou longueur ou coût) associé à celle-ci. Selon un mode de réalisation préféré du procédé selon la présente invention, le poids associé à une première transition (correspondant à l'hypothèse : "échantillon de données ordinaire reçu") d'un premier état vers un deuxième état est défini de façon à dépendre de la différence entre l'échantillon de données associé au deuxième état et un échantillon de données prédécesseur direct présumé dans le même canal de données que l'échantillon de données associé au deuxième état. De plus, le poids associé à une deuxième transition (correspondant à l'hypothèse "échantillon de données supplémentaire reçu") d'un premier état vers un deuxième état dépend de la différence entre un ou plusieurs des échantillons de données subséquents à l'échantillon de données associé au deuxième état et un échantillon de données prédécesseur direct présumé dans le même canal de données que les un ou plusieurs échantillons de données subséquents. Plus la différence est élevée, plus fort sera également le poids de la transition correspondante. Ce mode de réalisation du procédé exploite ainsi le fait que les dérivées locales de données provenant du même canal de données sont faibles, tandis que des dérivées inter-locales de données entre deux canaux de données distincts sont comparativement élevées. En conséquence, en l'absence de glitch, une transition du premier type obtiendra un poids faible tandis qu'une transition du deuxième type obtiendra un poids plus fort. De manière similaire, en présence d'un glitch, une transition du

premier type obtiendra un poids fort tandis qu'une transition du deuxième type obtiendra un poids plus faible.

[0009] L'exécution de l'algorithme de Viterbi comprend la construction d'une pluralité de chemins au travers des états, chaque chemin de la pluralité de chemins correspondant à une séquence spécifique de transitions des premier et deuxième types, chaque chemin de la pluralité de chemins possédant associé à celui-ci un poids cumulé (ou longueur ou coût) correspondant à la somme des poids associés aux première et deuxième transitions constituant le chemin, et la recherche d'un chemin de poids minimal parmi la pluralité de chemins. Puisque l'on est intéressé par la recherche du chemin possédant le poids cumulé le plus faible, à chaque croisement de deux chemins, le chemin avec le poids cumulé le plus fort est rejeté. Chaque chemin de la pluralité de chemins possède de préférence associé à celui-ci une séquence d'échantillons de données présumés ordinaires. Le flux corrigé de données multiplexées correspond alors à la séquence d'échantillons de données associée au chemin de poids minimal trouvé.

[0010] Selon un mode de réalisation préféré du procédé selon l'invention, avant l'exécution de l'algorithme de Viterbi, des trames susceptibles de contenir au moins un échantillon de données supplémentaire sont identifiées. Les poids associés aux première et deuxième transitions d'un premier état vers un deuxième état peuvent dans ce cas être modifiés (re-pondérés) selon que l'échantillon de données associé au deuxième état appartient à une trame identifiée comme étant susceptible de contenir au moins un échantillon de données supplémentaire.

[0011] Un aspect préféré de la présente invention porte sur un programme informatique comprenant des instructions pouvant être mises en œuvre par un processeur, qui, lorsqu'elles sont exécutées par un processeur, amènent le processeur à exécuter un procédé tel que celui décrit ici. Un autre aspect préféré de la présente invention concerne un dispositif informatique comprenant un processeur et une mémoire de préférence non volatile contenant en mémoire des instructions pouvant être mises en œuvre par un processeur, qui,

lorsqu'elles sont exécutées par le processeur, amènent le processeur à exécuter un procédé tel que celui décrit ici.

Description succincte des schémas

[0012] Des modes de réalisation préférés de la présente invention sont décrits ci-après, à titre d'exemple, en faisant référence aux schémas annexés dans lesquels :

La Figure 1 est une illustration schématique d'un flux de données multiplexées contenant un échantillon de données supplémentaire qui amène des échantillons subséquents à être décalés par rapport à leurs intervalles attribués,

Les Figures 2 à 4 sont des illustrations schématiques de la construction d'un treillis au moyen de l'algorithme de Viterbi dans un procédé selon un mode de réalisation préféré de la présente invention,

La Figure 5 illustre la performance du procédé selon la présente invention.

Description de modes de réalisation préférés

1. Introduction à MADRAS

[0013] Dans la suite, des modes de réalisation préférés de la présente invention seront illustrés par l'exemple de la détection et de la correction de glitchs dans des données provenant de MADRAS, qui est un imageur hyperfréquence embarqué sur le satellite MEGHA-TROPIQUES. Il convient de noter, cependant, que le procédé selon la présente invention peut faire l'objet d'autres applications.

[0014] Les données scientifiques acquises par MADRAS se composent de plusieurs centaines d'acquisitions (ou balayages) enregistrés dans $N_c = 11$ canaux. Un ensemble de balayages contigus compose une image multicanal. Un balayage donné de 11 canaux résulte du réordonnancement d'un flux de données unique. Chaque cycle de N_c échantillons de données correspond à un pixel de l'image et est appelé une trame. Le flux de données contient ainsi une séquence cyclique de $11^{\circ}N_F$ mesures qui sont séquentiellement et périodiquement attribuées aux 11 canaux, où N_F est le nombre de trames. Du

fait d'un dysfonctionnement dans la chaîne de communication entre deux dispositifs électroniques, ce flux de données est corrompu.

[0015] Le mécanisme de corruption le plus fréquent est tel que des échantillons de données additionnels (glitchs) sont concaténés aux mesures du fait de l'occurrence de signaux à impulsion parasites (dans le contexte de MADRAS, cette anomalie est appelée "anomalie de type 1"). En conséquence, les données collectées contiennent la plus large part des mesures scientifiques, mais les insertions de données multiples et aléatoires résultent en

- la présence d'échantillons de données supplémentaires (les glitchs) dans le flux de données,

- des permutations cycliques des canaux après recombinaison du flux de données du fait de ces données supplémentaires.

[0016] Une vue schématique du phénomène est illustré à la Figure 1, où, à des fins de concision, seulement quatre canaux ont été considérés (c'est-à-dire N_c = 4 à la Figure 1) et le flux de données **10** est divisé en trois trames successives **12**. Chaque canal de données possède son intervalle attribué à l'intérieur de chaque trame. À la Figure 1, les nombres **1** à **4** désignent les échantillons de données appartenant aux différents canaux de données. Lorsqu'un glitch (appelé **G**) apparaît dans l'intervalle temporel réservé pour le canal **3**, cet intervalle reçoit alors une valeur aberrante (qui ne peut pas être attribuée à l'un quelconque des canaux de données), l'intervalle subséquent réservé pour le canal **2** reçoit l'échantillon de données appartenant au canal **3**, l'intervalle réservé suivant pour le canal **1** reçoit l'échantillon de données appartenant au canal **2**, et le décalage (discordance entre canaux et intervalles) est propagé vers la trame suivante.

[0017] L'algorithme décrit ci-dessous est conçu de façon à détecter et supprimer ces glitchs pour chaque balayage individuellement. Une fois que tous les glitchs ont été supprimés d'un balayage, les données contiennent uniquement les mesures qui sont récupérables, chaque échantillon de données étant situé dans son intervalle dédié sur l'ensemble des trames. Les données corrigées sont en conséquence scientifiquement exploitables.

[0018] Un deuxième mécanisme de corruption fréquent (dit "anomalie de type 3") est caractérisé par l'observation de valeurs erratiques au cours de plusieurs trames au début du balayage suivies par une valeur unique constante dans tous les canaux pour le reste du balayage. Si une anomalie de type 3 est détectée, la totalité du balayage doit être rejetée. Les anomalies de type 3 et autres types moins fréquents d'anomalies, ne seront plus abordés dans le présent document.

[0019] Selon la présente invention, l'algorithme de Viterbi est utilisé pour la détection et la suppression de glitchs dans le signal multicanal. L'algorithme de Viterbi est utilisé pour trouver le chemin avec un poids cumulé minimal au travers d'un treillis approprié. Pour l'application de l'algorithme de Viterbi, un treillis et les distances (poids) associées à toutes les branches de ce treillis sont définis. Un treillis est un graphe orienté dont les nœuds sont organisés en tranches verticales (au niveau du même incrément temporel discret) et chaque nœud est raccordé à au moins un nœud au niveau de l'incrément temporel précédent et au moins un nœud au niveau de l'incrément temporel suivant. Étant donnée une fonction de coût spécifique, la solution donnée par l'algorithme est le chemin optimal au travers du treillis, c'est-à-dire le chemin qui possède le coût le plus faible parmi l'ensemble de tous les chemins possibles au travers du treillis du premier instant au dernier instant. Dans l'application proposée, le chemin solution indique les emplacements de glitchs, permettant ainsi leur correction.

2. Détection et correction d'anomalies de Type 1

[0020] Comme indiqué ci-dessus, étant donné une fonction de coût spécifique, la solution donnée par l'algorithme de Viterbi est optimale dans le sens où le chemin solution possède le coût le plus faible parmi tous les chemins possibles au travers du treillis à partir d'un instant de départ (instant 0) à un instant terminal. L'application de l'algorithme de Viterbi à un problème spécifique exige par conséquent la conception d'un treillis et la spécification d'une métrique appropriée pour ses branches. Une mise en œuvre possible de l'algorithme de Viterbi de détection et de suppression de glitchs est donnée ci-

après. D'autres détails supplémentaires peuvent être trouvés dans les pseudocodes informatiques à la fin de cette description. Dans la suite, x(n) désignera l'échantillon de données reçu à l'instant discret n. x(n) peut être un échantillon de données ordinaire ou un échantillon de données supplémentaire (glitch). L'ensemble de tous les échantillons reçus séquentiellement x(n) (mesures et glitchs) représente le flux de données.

2.1. Conception du treillis

[0021] En tant qu'états de l'algorithme de Viterbi, on utilise les décalages possibles (c'est-à-dire un décalage d'un intervalle, deux intervalles, etc.) provoqués par des échantillons de données supplémentaires reçus jusqu'à l'échantillon de données associé. Le treillis contient N_s ($\geq N_c$) nœuds par instant n (c'est-à-dire arrivée d'un échantillon) qui sont désignés par $c_{k;n}$, avec k = 0, ..., N_s - 1 à des fins de commodité. Le nœud $c_{k;n}$ correspond à la valeur d'état dans laquelle k (modulo N_s - 1) glitchs ont été détectés et supprimés le long du chemin l'atteignant (c'est-à-dire à partir des n échantillons reçus jusque là).

[0022] Chaque nœud $c_{k;n}$ est raccordé aux nœuds $c_{k-1;n-1}$ et $c_{k;n-1}$, associés à l'instant précédent, et aux nœuds $c_{k;n+1}$ et $c_{k+1;n+1}$, associés à l'instant suivant. Le treillis est "circulaire" dans le sens où le nœud $c_{0;n}$ est raccordé au nœud $c_{Ns-1;n-1}$, et le nœud $c_{Ns-1;n}$ est raccordé au nœud $c_{0;n+1}$. Les sommets raccordant les nœuds sont appelés branches. La branche raccordant les nœuds $c_{k;n}$ et $c_{k;n-1}$ est désignée par $v^0_{k;n}$, et la branche raccordant les nœuds $c_{k;n}$ et $c_{k-1;n-1}$ est désignée par $v^1_{k;n}$. Notez que la branche $v^1_{k;n}$ correspond à l'hypothèse "x(n) est un glitch" pour l'état k, tandis que la branche $v^0_{k;n}$ correspond à l'hypothèse "x(n) n'est pas un glitch" pour l'état k.

2.2. Algorithme de Viterbi

[0023] Chaque branche $v^0_{k;n}$ et $v^1_{k;n}$ possède un poids d_0(k; n) et un poids d_1(k; n) qui lui sont attribués. Une définition possible des poids d_0(k; n) et d_1(k; n) est décrite dans la sous-section qui suit. En résumé, l'algorithme de Viterbi supprime à chaque instant n toutes sauf une des branches atteignant le même état $c_{n;k}$, de sorte que chaque état $c_{n;k}$ peut être atteint par uniquement un chemin unique au travers du treillis.

[0024] À l'instant n-1, à chaque nœud a été attribué le poids cumulé D(k; n-1) des branches du chemin unique l'atteignant. $[\hat{x}]_{k;n-1}$ désigne la séquence d'échantillons vraisemblablement valides rencontrés sur le chemin unique atteignant le nœud $c_{k;n-1}$. Dans la suite, nous utiliserons la notation : $[\hat{x}]_{k;n-1} = [\hat{x}_k(1), \hat{x}_k(2), \ldots, \hat{x}_k(n_k)]$, où n_k désigne le nombre d'échantillons vraisemblablement valides reçus sur le chemin atteignant le nœud $c_{k;n-1}$. Notez que $n_k = n-k$ puisque k glitchs ont été détectés à l'état k.

[0025] Si au nœud $c_{k;n}$, la somme de D(k;n-1) et de $d_0(k; n)$ est inférieure à la somme de D(k-1; n1) et de $d_1(k; n)$, c'est-à-dire si l'hypothèse "x(n) n'est pas un glitch" conduit à un poids cumulé plus petit D(k; n) que l'hypothèse "x(n) est un glitch":

- La branche $v^1_{k;n}$ est supprimée du treillis et la branche $v^0_{k;n}$ est conservée.
- L'échantillon x(n) est accepté en tant que mesure valide (échantillon de données ordinaire) par l'état $c_{k;n}$. La séquence de tous les échantillons qui ont été acceptés le long du chemin unique atteignant le nœud $c_{k;n}$ est ainsi obtenue par $[\hat{x}]_{k;n} = [[\hat{x}]_{k;n-1}, x(n)]$, où [. , .] est l'opérateur de concaténation.

[0026] Dans le cas contraire, si au nœud $c_{k;n}$, la somme de D(k; n-1) et de $d_0(k; n)$ est supérieure à la somme de D(k-1; n-1) et de $d_1(k; n)$, c'est-à-dire si l'hypothèse "x(n) n'est pas un glitch" conduit à un poids cumulé plus fort D(k; n) que l'hypothèse "x(n) est un glitch":

- La branche $v^0_{k;n}$ est supprimée du treillis et la branche $v^1_{k;n}$ est conservée.
- L'échantillon x(n) n'est pas accepté en tant que mesure valide (échantillon de données ordinaire) par l'état $c_{k;n}$.
- La séquence résultante de mesures valides $[\hat{x}]_{k;n}$ sur le chemin atteignant le nœud $c_{k;n}$ est ainsi la même que la séquence de mesures valides atteignant le nœud $c_{k-1;n-1}$: $[\hat{x}]_{k;n} = [\hat{x}]_{k-1;n-1}$.

[0027] Après réception du dernier échantillon de données x(N), le chemin au travers du treillis avec le poids cumulé le plus faible D(k; N) est choisi.

Désigné par $\bar{k} = \arg \min_{k=0,\ldots,Ns-1} (D(k;N))$, le flux de données corrigé est donnée par la séquence $[\hat{x}]_{\bar{k};N}$..

2.3. Métrique de branche

[0028] La métrique attribuée aux branches de treillis est basée sur des dérivées locales d'un échantillon de données x(n) avec des échantillons de données reçus passés et futurs.

1) Le poids attribuée à la branche $v^0_{k;n}$ raccordant les nœuds $c_{k;n}$ et $c_{k;n-1}$ est donné par la racine carrée de la valeur absolue de la différence entre x(n) et le dernier échantillon de données valide qui est supposé appartenir au même canal de données que x(n) que l'état $c_{k;n}$ a reçu, à savoir $\hat{x}_k(n_k-N_c+1)$. En conséquence :

$$d_0(k,n) = \sqrt{|x(n) - \hat{x}_k(n_k - N_c + 1)|} \qquad \text{(Equation. 1)}$$

2) Le poids $d_1(k; n)$ attribué aux branches $v^1_{k;n}$ raccordant les nœuds $c_{k;n}$ et $c_{k-1;n-1}$, k = 0, ..., N_s-1 est construit sous la forme d'une moyenne d'un sous-ensemble des différences entre les derniers échantillons de données valides $\hat{x}_k(n_k-Nc+1)$ sur les chemins qui atteignent les états $c_{k;n}$, k = 0, ..., N_s-1, et un nombre N_{future} d'échantillons futurs x(n+1), x(n+2), ..., x(n+N_{future}). Le sous-ensemble de différences est donné par les $N_{future}/2$ différences les plus petites associées à chaque état. Dans le cas de MADRAS, le choix N_{future} = 10 s'est avéré être avantageux.

$$d_1(k,n) = \frac{WD_1}{N_s} \sum_{k=0}^{N_s-1} h(k,n) \text{, where} \qquad \text{(Equation. 2)}$$

$$h(k,n) = \frac{2}{N_{future}} \sum_{i=1}^{N_{future}/2} \delta_{k,n}(i) \text{, with} \qquad \text{(Equation. 3)}$$

$$\left[\delta_{k,n}(i) \,|\, i = 1,\ldots,N_{future}\right] = sort^{\uparrow}\left\{\sqrt{|x(n+j) - \hat{x}(n_k - N_c + 1|} \,\middle|\, j = 1,\ldots,N_{future}\right\} \qquad \text{(Equation. 4)}$$

Le coefficient WD_1 est un facteur de normalisation. Dans cet exemple : $WD_1 = \sqrt{\pi}$. Il convient de noter que les poids $d_1(k,n)$ ne dépendent pas de k, c'est-à-dire qu'ils sont identiques pour toutes les branches v^1.

[0029] Ce choix des poids peut être motivé en fonction de l'observation que les données dans chaque canal de données, considéré individuellement, présentent un comportement correct dans le sens où leur dérivée locale demeure à l'intérieur de certaines limites (qui dépendent du type de données). Le poids $d_0(k,n)$ correspond fondamentalement à la dérivée locale au niveau de l'échantillon x(n) si cet échantillon et le précédent ne sont pas des glitchs. En conséquence, $d_0(k,n)$ sera faible dans le sens où il ne sera pas supérieur à une variation normale dans un canal de données donné en l'absence de glitch. Les branches v^1 sont pondérées selon l'hypothèse que x(n) est un glitch. Si cette hypothèse est correcte, les données après x(n) sont décalées vers une position à l'intérieur de la trame par rapport à des données avant x(n). Avec la correction pour le décalage appliquée, on peut s'attendre à nouveau que les canaux de données présentent un comportement correct dans le sens ci-dessus. Le calcul de $d_1(k,n)$ est basé sur le calcul de quantités $\delta_{k,n}(i)$ correspondant aux dérivées locales pour un certain nombre de canaux de données (Équation 4), où il est supposé que x(n) lui-même doit être rejeté comme étant un glitch. Les quantités $\delta_{k,n}(i)$ sont triées dans un ordre ascendant et uniquement la moitié inférieure de celles-ci est utilisée pour calculer la quantité h(k,n) (Équation 3), qui représente une quantité qui peut être considérée comme une dérivée locale moyenne (où la moyenne est prise sur les canaux de données présentant le comportement le plus correct uniquement). Le fait de prendre la moyenne uniquement sur les quantités les plus faibles $\delta_{k,n}(i)$ réduit le risque que la quantité h(k,n) soit dénaturée du fait d'une variation naturelle importante dans un canal de données ou du fait d'un glitch se produisant peu de temps après x(n) dans le flux de données. $d_1(k,n)$ est finalement obtenu (Équation 2) par le calcul de la moyenne des quantités h(k,n), k = 0, ..., N_s-1, sur les états possibles (les différentes hypothèses pour le nombre de glitchs qui se sont produits à ce stade).

[0030] Les algorithmes de pseudocode 1 et 2 à la fin de la description mettent en œuvre l'algorithme de Viterbi décrit dans les sections 2.1. à 2.3.

[0031] Les Figures 2 à 4 illustrent comment fonctionne l'algorithme de Viterbi ci-dessus sur un flux de données avec quatre intervalles par trame. À des fins d'illustration, la métrique de branche est simplifiée par rapport à

l'exemple ci-dessus. Les poids des branches sont indiqués dans le tableau ci-après :

branches	pas de glitch	glitch
i glitch → i glitch	0 (dans l'état correct) 2 (dans l'état erroné)	3
i glitch → $(i+1)_{mod5}$ glitch	3	1

[0032] Les branches $v^1_{k;n}$ (correspondant à l'hypothèse "x(n) est un glitch") reçoivent

- le poids 1 si l'échantillon x(n) est effectivement un glitch et
- le poids 3 si l'échantillon x(n) est en réalité un échantillon de données ordinaire.

Les branches $v^0_{k;n}$ (correspondant à l'hypothèse "x(n) n'est pas un glitch") reçoivent

- le poids 0 si l'échantillon x(n) est un échantillon de données ordinaire et l'état correspondant est l'état correct,
- le poids 2 si l'échantillon x(n) est un échantillon de données ordinaire mais l'état correspondant est erroné, et
- le poids 3 si l'échantillon x(n) est en réalité un glitch.

Il convient de noter que la métrique utilisée ici est artificielle parce que dans une situation de la vie réelle, on n'aurait pas de connaissance a priori de la dynamique de chaque canal de données. À l'instant n=0, une trame entière a été reçue.

[0033] La Figure 2 illustre le treillis après la réception de l'échantillon de données (ordinaire) à l'instant n=1. Le poids de chaque branche individuelle est indiqué au dessus de la branche respective. Les poids cumulés sont illustrés au-dessus des nœuds du treillis. Le poids de la branche $v^0_{0,1}$ est déterminé par

la comparaison de l'échantillon de données x(n=1) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité de la branche $v^0_{0,1}$ est $c_{0;1}$, aucun glitch n'est considéré s'être produit à ce stade et l'échantillon de données x(n=1) est donc comparé à l'échantillon de données N_c (dans ce cas 4) intervalles de temps vers l'avant, c'est-à-dire avec x(n=-3). Puisque x(n=1) et x(n=-3) appartiennent en réalité au même canal de données, le poids $d_0(0,1)$, calculé au moyen de l'équation 1 ci-dessus, sera comparativement faible. Dans le cas illustré, ce fait est reflété par le fait que la branche $v^0_{0,1}$ reçoit le poids 0, selon le tableau ci-dessus. Le poids de la branche $v^1_{1,1}$ est déterminé au moyen de l'équation 2 ci-dessus. L'hypothèse est ici que x(n=1) est un glitch mais, comme x(n=1) n'est pas un glitch, le poids $d_1(1,1)$ sera comparativement fort. Dans le cas illustré, ce fait est reflété par le fait que la branche $v^1_{1,1}$ reçoit le poids 3.

[0034] La Figure 3 illustre le treillis après la réception de l'échantillon de données (supplémentaire) à l'instant n=2. Le poids de la branche $v^0_{0,2}$ est déterminé par la comparaison de l'échantillon de données x(n=2) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité de la branche $v^0_{0,2}$ est $c_{0;2}$, aucun glitch n'est considéré s'être produit à ce stade et l'échantillon de données x(n=2) est donc comparé à l'échantillon de données quatre intervalles de temps vers l'avant, c'est-à-dire à x(n=-2). Puisque x(n=2) est en réalité un glitch, le poids $d_0(0,2)$ sera comparativement fort. Dans le cas illustré, ce fait est reflété par le fait que la branche $v^0_{0,2}$ reçoit le poids 3. Le poids de toutes les branches $v^1_{k,2}$ (k=0, 1, 2, ...) est déterminé au moyen de l'équation 2 ci-dessus. L'hypothèse est que x(n=2) est un glitch et puisque ceci est effectivement le cas, les poids $d_1(k,2)$ seront comparativement faibles. Dans le cas illustré, les branches $v^1_{1,2}$ et $v^1_{2,2}$ reçoivent le poids 1. Le poids de la branche $v^0_{1,2}$ est déterminé au moyen de l'équation 1, c'est-à-dire par la comparaison de l'échantillon de données x(n=2) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité est $c_{1;2}$, un glitch est considéré s'être produit à ce stade, l'échantillon de données x(n=2) est donc comparé à l'échantillon de données N_c+1 (ici : 5) intervalles de temps vers l'avant, c'est-à-dire à x(n=-3). Puisque

x(n=2) est en réalité un glitch, le poids $d_0(1,2)$ sera comparativement fort (poids 3). Le nœud $c_{1,2}$ pourrait maintenant être atteint par deux chemins différents. L'algorithme de Viterbi supprime ainsi le chemin avec le poids cumulé le plus fort.

[0035] La Figure 4 illustre le treillis après la réception de l'échantillon de données (ordinaire) à l'instant n=3. Le poids de la branche $v^0_{0,3}$ est déterminé par la comparaison de l'échantillon de données x(n=3) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité de la branche $v^0_{0,3}$ est $c_{0;3}$, aucun glitch n'est considéré s'être produit à ce stade et l'échantillon de données x(n=3) est donc comparé à l'échantillon de données quatre intervalles de temps vers l'avant, c'est-à-dire à x(n=-1). Puisqu'un glitch s'est produit entre temps à n=2, il existe une discordance de canal non prise en compte et le poids $d_0(0,3)$ sera comparativement fort mais inférieur au poids que l'on obtiendrait par la comparaison d'un glitch à un échantillon de données ordinaire (poids 2). Le poids de toutes les branches $v^1_{k,3}$ (k=0, 1, 2, ...) est déterminé au moyen de l'équation 2 ci-dessus. L'hypothèse est que x(n=3) est un glitch, mais puisque x(n=3) n'est pas en réalité un glitch, les poids $d_1(k,3)$ seront comparativement forts. Les branches $v^1_{1,3}$, $v^1_{2,3}$ et $v^1_{3,3}$ reçoivent donc le poids 3. Le poids de la branche $v^0_{1,3}$ est déterminé au moyen de l'équation 1 par la comparaison de l'échantillon de données x(n=3) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité de la branche $v^0_{1,3}$ est $c_{1;3}$, un glitch est considéré s'être produit à ce stade. L'échantillon de données x(n=3) est donc comparé à l'échantillon de données N_c+1 intervalles de temps vers l'avant, c'est-à-dire à x(n=-2). Puisque x(n=-2) et x(n=3) appartiennent en réalité au même canal de données, le poids $d_0(1,3)$ sera comparativement faible (poids 0). Le poids de la branche $v^0_{2,3}$ est également déterminé au moyen de l'équation 1 par la comparaison de l'échantillon de données x(n=3) à son prédécesseur présumé dans le même canal de données. Puisque le nœud d'extrémité de la branche $v^0_{2,3}$ est $c_{2;3}$, deux glitchs sont considérés s'être produits à ce stade. L'échantillon de données x(n=3) est donc comparé à l'échantillon de données N_C+2 (ici : 6) intervalles de temps vers l'avant, c'est-à-dire à x(n=-3). Puisque x(n=-3) et

x(n=3) n'appartiennent pas en réalité au même canal de données, le poids $d_0(2,3)$ sera comparativement fort (poids 2). Chacun des nœuds $c_{1,3}$ et $c_{2,3}$ pourrait maintenant être atteint par deux chemins différents. L'algorithme de Viterbi supprime chaque fois le chemin avec le poids cumulé le plus fort.

[0036] Pendant que le treillis est construit, l'algorithme conserve une trace des séquences $[\hat{x}]_{k;n}$ de échantillons de données vraisemblablement valides. À l'instant n=2 :

- $[\hat{x}]_{0;2}$ = [x(n=1), x(n=2)],
- $[\hat{x}]_{1;2}$ = [x(n=1)],
- o $[\hat{x}]_{2;2}$ = [ ] (séquence vide).

A l'instant n=3 :

- $[\hat{x}]_{0;3}$ = [x(n=1), x(n=2), x(n=3)], obtenu en tant que concaténation de [$[\hat{x}]_{0;2}$, x(n=3)],
- $[\hat{x}]_{1;3}$ = [x(n=1), x(n=3)], obtenu en tant que concaténation de [$[\hat{x}]_{1;2}$, x(n=3)],
- $[\hat{x}]_{2;3}$ = [x(n=1)], obtenu par la reprise de la séquence du nœud précédent sur le chemin : $[\hat{x}]_{2;3} = [\hat{x}]_{1;2}$
- o $[\hat{x}]_{3;3}$ = [ ] (séquence vide), obtenu par la reprise de la séquence du nœud précédent sur le chemin: $[\hat{x}]_{3;3} = [\hat{x}]_{2;2}$.

[0037] Lorsque le dernier échantillon x(N) a été reçu, l'algorithme de Viterbi sélectionne le nœud $c_{k,N}$, k = 0, 1, ..., N_s, sur lequel le chemin avec le poids cumulé le plus faible arrive. Supposons que $c_{kmin,N}$ soit ce nœud, le flux de données corrigé peut être produit sous la forme $[\hat{x}]_{kmin,N}$. Par exemple, si x(n=3) était le dernier échantillon reçu, on trouverait en tant que flux de données corrigé la séquence $[\hat{x}]_{1;3}$ = [x(n=1), x(n=3)].

3. Détection de trames corrompues

[0038] Cette section décrit un algorithme qui permet la détection de trames qui potentiellement contiennent des anomalies de type 1. Similaire à

l'algorithme décrit dans la section précédente 2, il utilise une métrique basée sur des dérivées locales afin de décider si une trame est corrompu ou non. Cependant, la métrique est construite conjointement pour tous les canaux dans une trame, ce qui ajoute de la robustesse contre les fausses alarmes dans des zones du balayage où les N_c canaux prennent des valeurs qui sont très proches les unes des autres. Il est conçu de façon à travailler trame par trame et peut en conséquence être uniquement utilisé pour détecter si une trame contient un glitch ou non et ne peut pas résoudre le problème de savoir si un échantillon individuel est un glitch ou non.

3.1. Métrique et critère de détection

[0039] Soit m l'indice des trames et désignons par $x_m^{(i)}$ les échantillons dans cette trame, i = 1, ..., N_c. L'algorithme est basé sur la métrique double suivante :

1) différences intra-canaux (entre échantillons du même canal de données i) :

$$\delta_m(i) = x_{m+1}^{(i)} - x_m^{(i)}$$

$$D_m = \sqrt{\sum_{i=1}^{Nc} |\delta_m(i)|^2}$$

2) différences inter-canaux (entre échantillons du canal i et du canal (i+d_i) modulo N_c, où d_i = 1, ..., N_c-1) :

$$\delta_m(i, d_i) = x_{m+2}^{([i+d_i-1])} - x_m^{(i)}, \text{ où } [.] = \text{mod}(., N_c)$$

$$T_m(d_i) = \sqrt{\sum_{i=1}^{Nc} |\delta_m(i, d_i)|^2}$$

[0040] Le principe qui sous-tend cette métrique réside dans les observations suivantes :

- Si la trame m et la trame m + 1 ne contiennent pas de glitchs, alors les différences intra-canaux $\delta_m(i)$ correspondent aux dérivées locales des mesures qui sont transmises dans le canal i, et les différences inter-canaux $\delta_m(i,d_i)$ jusqu'à la différence des mesures transmises dans l'intervalle i et l'intervalle i + d_i. La somme des

différences intra-canaux sera faible comparée à la somme des différences inter-canaux.

- Si la trame m ou la trame m + 1 contient n_g glitchs, alors les différences intra-canaux $\delta_m(i)$ correspondent soit à la différence des mesures transmises dans l'intervalle i et les mesures qui auraient dû être transmises dans l'intervalle j = i + n_g, ou à la différence des mesures transmises dans le canal i et la valeur du glitch. Dans les deux cas, la somme des différences intra-canaux sera élevée. A l'inverse, les différences inter-canaux $\delta_m(i,d_i)$ qui satisfont la condition $d_i = n_g$ correspondent aux dérivées locales des mesures qui doivent être transmises dans le canal i. Leur somme sera en conséquence faible.

[0041] En conséquence, des glitchs dans une trame m* sont considérés détectés si le critère suivant est satisfait :

$$D_{m^*} > \min_{d_i} T_{m^*}(d_i)$$

Le nombre de glitchs dans la trame m* peut être estimé par :

$$n_g(m^*) > \arg\min_{d_i} T_{m^*}(d_i)$$

3.2. Algorithme pour la détection de trames corrompues

[0042] L'algorithme (algorithmes de pseudocode 3 à 5 à la fin de la présente description) boucle au travers des trames m d'un balayage. Si pour la trame m*, le critère $D_{m^*} > \min_{d_i} T_{m^*}(d_i)$ est satisfait et que par conséquent des glitchs sont détectés :

1) La trame m* est détectée comme étant corrompue et l'indice de la trame est placé en mémoire.

2) La trame m* est supprimée des données. Plus précisément, la trame m* (qui est présumée contenir au moins un glitch) est rejetée, et la

trame m*+1 prend sa place, c'est-à-dire, est maintenant à la position m* pour la suite de l'algorithme de détection de trames corrompues.

3) Les canaux de toutes les trames m* + 1, m* + 2, ... sont permutés de manière circulaire par $n_g(m^*)$. En conséquence, $x_{m^*-1}^{(l)}$ et $x_{m^*+1}^{(l)}$ contiennent des mesures provenant du même canal de données et l'ordre des canaux a été rétabli.

4) L'algorithme poursuit avec la trame suivante du flux de données.

3.3. Algorithme de Viterbi pondéré combiné à la détection de trames corrompues

[0043] Dans cette section, l'algorithme de Viterbi décrit dans la section 2 est généralisé afin de permettre l'utilisation d'informations antérieures sur un échantillon spécifique x(n). Cette généralisation est motivée par le fait que, dans de nombreux cas, la plupart des échantillons dans un balayage sont en réalité des mesures valides. Si, par exemple, des séquences d'échantillons sont connues être des mesures valides (ou, au contraire, des glitchs), cette information peut être utilisée pour améliorer les performances de l'algorithme de Viterbi.

[0044] Les informations antérieures placent l'algorithme sous la forme d'un facteur γ(n) qui agit en tant que multiplicateur pour les poids $d_1(k, n)$ donnés par l'équation 2. Au lieu de $d_1(k, n)$, on utilise alors $\tilde{d}_1(k,n)$ qui est défini sous la forme :

$$\tilde{d}_1(k,n) = \gamma(n) \cdot d_1(k,n) \,.$$

[0045] En conséquence, le facteur γ(n) influence la décision selon laquelle les branches $v^0_{k;n}$ ou les branches $v^1_{k;n}$ doivent être supprimées à l'instant n.

- Cas $\gamma(n) = \infty$

  Le choix $\gamma(n) = \infty$ est fait si l'échantillon x(n) est connu être une mesure valide (un échantillon de données ordinaire). En fonction de cette information antérieure, l'algorithme de Viterbi est forcé de supprimer les

branches v1k;n du treillis (pour tous les k) et d'accepter l'échantillon x(n) en tant que mesure valide.

- Cas $1 < \gamma(n) < \infty$

Ce choix est fait si l'échantillon x(n) est très probablement une mesure valide (un échantillon de données ordinaire). Les branches v1k;n sont pénalisées par leur accroissement en termes de poids. En conséquence, la probabilité que x(n) soit accepté en tant qu'échantillon valide est augmentée (en comparaison de l'algorithme qui utilise d1(k, n) au lieu de $\tilde{d}_1(k,n)$).

- Cas $\gamma(n) = 0$

Ce choix est fait s'il est connu que l'échantillon x(n) est un glitch. Le fait que les branches v1k;n recevront un poids nul amène l'algorithme de Viterbi à supprimer les branches v0k;n du treillis. L'échantillon x(n) est détecté comme étant un glitch et n'est pas accepté en tant qu'échantillon valide par aucun état ck;n, quelle que soit la valeur de k.

- Cas $0 < \gamma(n) < 1$

Ce choix est fait s'il est très probable que l'échantillon x(n) soit un glitch. Le poids des branches $v^1_{k;n}$ est diminué comparé à l'algorithme qui utilise $d_1(k, n)$. En conséquence, la probabilité que x(n) soit accepté en tant qu'échantillon valide est diminuée.

[0046] L'algorithme de Viterbi pondéré est détaillé dans les algorithmes de pseudocode 6 et 7 à la fin de la description. Il est couplé ici à un algorithme de détection de trames corrompues, par exemple, l'algorithme détaillé à la section 3.2. (algorithmes de pseudocode 3 à 5) avec Mweight = infini (∞). L'algorithme de pseudocode 6 attribue en conséquence un poids $\gamma(n)$ = 1 à tous les échantillons dans des trames qui sont détectées comme étant corrompues et le poids $\gamma(n) = \infty$ à tous les échantillons dans des trames qui ne sont pas détectées comme étant corrompues. The branches $v^1_{k;n}$ sont par conséquent supprimées du treillis des algorithmes de pseudocode 6 et 7 pour tous les

échantillons dans des trames qui ne sont pas détectées comme étant corrompues par les algorithmes de pseudocode de détection de trames corrompues 3 à 5. En conséquence, le coût de traitement de l'algorithme de Viterbi est réduit pour des balayages qui sont uniquement faiblement corrompus. De plus, la probabilité de fausse alarme (c'est-à-dire la détection d'un glitch non existant et la suppression d'une mesure valide) est diminuée, en particulier dans des zones où tous les canaux possèdent une dynamique très similaire.

[0047] La Figure 5 illustre la performance de l'algorithme de détection et suppression de glitchs selon les algorithmes de pseudocode 6 et 7 sur un ensemble de données synthétiques. Le graphe **14** illustre les données d'origine. Chaque instant (axe x) correspond à une trame. Les données d'origine ont été artificiellement corrompues par l'insertion de glitchs (graphe 16), résultant en une permutation cyclique des canaux de données après chaque glitch. Les données corrompues du graphe **16** ont ensuite été introduites dans le procédé de détection et de suppression de glitchs. Le résultat de la correction est illustré sous la forme du graphe **18**. Comme on peut le constater, la correction fonctionne bien : les glitchs sont effectivement supprimés du flux de données corrigé. Il convient de noter que la correction est uniquement appliquée aux trames 37 à 485, dans lesquelles tous les canaux de l'instrument MADRAS contiennent des mesures.

[0048] Bien que des modes de réalisation spécifiques aient été décrits dans le détail, les personnes du métier pourront observer que diverses modifications et variantes à ces détails pourraient être élaborées à la lumière des conclusions globales de l'invention. En conséquence, les agencements particuliers décrits sont destinés à être uniquement illustratifs et non limitatifs en ce qui concerne la portée de la présente invention à laquelle il convient d'attribuer l'étendue complète des revendications annexées et de tous équivalents de celles-ci.

Algorithme 1 Algorithme de Viterbi Partie 1: Initialisation

——Entrée

$CorruptedScan$: $matrix\ [11 \times N_f]$ ▷ balayage corrompu – trames 37 à 485

____Sortie

$CorrectedScan$: $matrix\ [11 \times N_f]$ ▷ balayage corrigé – trames 37 à 485

$cweight$: ▷ distance cumulée du chemin du balayage corrigé

%% ——INITIALISATION —— %%

$N_c \leftarrow 11$; $N_s \leftarrow 2N_c$ ▷ CONST: #canaux ; #états de l'algorithme de Viterbi

$WD_1 \leftarrow \sqrt{\pi}$ ▷ CONST: multiplicateur de mesure de branche

$FUTUR \leftarrow 10$ ▷ CONST: #échantillons futurs dans la métrique de branche

$d_0(1:N_s) \leftarrow 0$; $d_1(1:N_s) \leftarrow 0$ ▷ distances de branche de treillis

$distances\_cum(1:N_s) \leftarrow \infty$; $distances\_cum(1) \leftarrow 1$ ▷ distance cumulée pour chaque état

$last\_index(1:N_s) \leftarrow N_s$ ▷ indice du dernier échantillon valide reçu

for $i = 1 \rightarrow N_c$ **do** ▷ placer le balayage dans le format vecteur et initialiser les données en sortie

  for $j = 1 \rightarrow N_f$ **do**

    $data((j-1) \cdot N_f + i) \leftarrow CorruptedScan(i,j)$

    $CorrectedScan(i,j) \leftarrow 0$

  end for

end for

$sample\_received(1:(N_f \cdot N_c), 1:N_s) \leftarrow 0$ ▷ Initialiser les vecteurs de chemin d'état

for $k = 1 \rightarrow N_s$ **do**

  for $n = 1 \rightarrow N_c$ **do**

    $sample\_received(n,k) \leftarrow scan(n)$

  end for

end for

$dd(1:FUTUR) \leftarrow 0$; $pre\_k \leftarrow 0$; $dist_0 \leftarrow 0$; $dist_1 \leftarrow 0$; $sid \leftarrow 0$; $cweight \leftarrow 0$ ▷ variables de travail et mémoires tampons

$old\_sample\_received(1:(N_f \cdot N_c), 1:N_s) \leftarrow 0$

$old\_distances\_cum(1:N_s) \leftarrow 0$

$old\_last\_index(1:N_s) \leftarrow 0$

... suite à l'algorithme 2

Algorithme 2 Algorithme de Viterbi Partie 2: boucle principale

... suite de l'algorithme 1

%% —— PRINCIPAL : BOUCLER SUR LES ÉCHANTILLONS —— %%

for $n = N_c + 1 \rightarrow N_c \cdot N_f - FUTUR$ **do**

  for $k = 1 \rightarrow N_s$ **do** ▷ **calculer la métrique de branche de l'échantillon *n* pour chaque état *k***

    $d_0(k) \leftarrow \mathbf{abs}(data(n) - sample\_received(k, last\_index(k) - N_c + 1))^{\frac{1}{2}}$

    $dd(1:FUTUR) \leftarrow \mathbf{abs}(data(n+1:n+FUTUR) - sample\_received(k, last\_index(k) - N_c + 1))^{\frac{1}{2}}$

    $dd \leftarrow \mathbf{sort}(dd)$ ▷ **trier dans l'ordre ascendant**

    $d_1(k) \leftarrow 0$

    for $j = 1 \rightarrow FUTUR/2$ **do**

      $d_1(k) = d_1(k) + WD_1 \cdot dd(j)/FUTUR$

    end for

  end for

  $old\_distances\_cum \leftarrow distances\_cum$

  $old\_last\_index \leftarrow last\_index$

  $old\_sample\_received \leftarrow sample\_received$

  for $k = 1 \rightarrow N_s$ **do** ▷ **actualiser les vecteurs d'état**

    $pre\_k \leftarrow k - 1$

    If $k == 0$ **then**

      $pre\_k \leftarrow N$

    end If

    $dist_0 \leftarrow old\_distances\_cum(k) + d_0(k)$ ▷ **distance de branche *k* → *k***

    $dist_1 \leftarrow old\_distances\_cum(pre\_k) + \mathbf{mean}(d_1)$ ▷ **distance de branche *k-1* → *k***

    If $dist_1 < dist_0$ **then** ▷ **échantillon considéré comme étant un glitch**

      $distances\_cum(k) \leftarrow dist_1$

      $sample\_received(:, k) \leftarrow old\_sample\_received(:, pre\_k)$

      $last\_index(k) \leftarrow old\_last\_index(pre\_k)$

    else ▷ **échantillon considéré comme étant valide**

      $distances\_cum(k) \leftarrow dist_0$

      $last\_index(k) \leftarrow last\_index(k) + 1$

      $sample\_received(last\_index(k), k) \leftarrow data(n)$

    end If

  end for

end for

%% —— DECISION —— %%

$sid \leftarrow \mathbf{index\_of\_min}(distances\_cum)$ ▷ **chemin avec la distance cumulée la plus faible**

$cweight \leftarrow distances\_cum(sid)$ ▷ **distance cumulée correspondante**

for $i = 1 \rightarrow N_c$ **do**

  for $j = 1 \rightarrow N_f$ **do**

    $CorrectedScan(i, j) \leftarrow sample\_received((j-1) \cdot N_f + i, sid)$ ▷ **placer le balayage corrigé dans un format matriciel**

  end for

end for

SORTIE $CorrectedScan$; $cweight$

Algorithme 3 Algorithme « Prototype » Récursif

——Entrée

$CorruptedScan$: $matrix\ [11 \times N_f]$ ▷ balayage corrompu – trames 37 à 485
$Mweight$: $CONST$ ▷ poids à attribuer à des échantillons dans des trames non corrompues

——Sortie

$Fvalid$: $vector\ [1 \times N_f]$ ▷ balises pour des trames non corrompues
$Gmetric$: $vector\ [1 \times 11 \cdot N_f]$ ▷ poids pour chaque échantillon dans *CorruptedScan*

%% ——INITIALISATION —— %%

$N_c \leftarrow 11$ ▷ CONST: #canaux ; #états de l'algorithme de Viterbi
$N_G \leftarrow N_c - 1$ ▷ CONST: ordre maximum de rotation de canaux
$DD \leftarrow 3$ ▷ CONST: voisinage de trame corrompue à baliser
$MITER \leftarrow 30$ ▷ CONST: nombre maximal d'itérations
$N_r \leftarrow N_f$
for $n = 1 \rightarrow N_f$ **do** ▷ indice de trames supposées non corrompues
  $NGindex(n) \leftarrow n$
  $Fvalid(n) \leftarrow 1$ ▷ balises pour trames non corrompues
  for $i = 1 \rightarrow N_c$ **do**
    $Gmetric((n-1) \cdot N_f + i) \leftarrow 1$ ▷ poids pour chaque échantillon
  end for
end for

%% —— RÉCURSION : DÉTECTION ET SUPPRESSION DE TRAMES CORROMPUES —— %%

$RUN \leftarrow 1$; $ITER \leftarrow 0$
$data\_r \leftarrow CorruptedScan$
while RUN==1 **do**
  $ITER \leftarrow ITER + 1$
  ← `DetectRemove` (cf. Algorithm 5)
end while

%% ——BALISER LES TRAMES CORROMPUES ET ATTTRIBUER DES POIDS POUR VITERBI —— %%

← `TagWeight` (cf. Algorithm 6)
SORTIE $Fvalid$, $Gmetric$

Algorithme 4 Algorithme « Prototype » Récursif: Partie DetectRemove

```
DetectRemove
%% ——— DÉTECTION DE TRAMES CORROMPUES ——— %%
iG ← 1                                              ▷ initialiser les compteurs pour cette récursion
TabG(1) ← 0;  PermG(1) ← 0
for n = 1 → N_r − 3 do                              ▷ Boucler sur les trames: Détection
                                                    ▷ calculer la distance i → i
    d_0 ← 0
    for i = 1 → N_c do
        d_0 ← d_0 + abs(data_r(i, n) − data_r(i, n + 1))^ν
    end for
    d_0 ← (d_0)^(1/ν)
    d_1(1 : N_G) ← 0;  d_P(1 : N_G) ← 0;  data_c(1 : N_c) ← data_r(1 : N_c, n)
    for j = 1 → N_G do                              ▷ calculer la distance i → i + j
        tmp ← data_c(1);  data_c(1 : N_c − 1) ← data_c(2 : N_c);  data_c(N_c) ← tmp
        for i = 1 → N_c do
            d_1(j) ← d_1(j) + abs(data_c(i, n) − data_r(i, n + 2))^ν
            d_P(j) ← d_P(j) + abs(data_c(i, n) − data_r(i, n + 3))^ν
        end for
        d_1(j) ← (d_1(j))^(1/ν);  d_P(j) = (d_P(j))^(1/ν)
    end for
    id_1 ← index_of_min(d_1);  id_P ← index_of_min(d_P)
    if d_0 > d_1(id_1) AND TabG(iG) < n then        ▷ La trame n + 1 présente un problème
        iG ← iG + 1;  TabG(iG) ← n + 1  PermG(iG) ← id_P
    end if
end for
iG ← iG − 1;
for i = 1 → iG do
    TabG(iG) ← TabG(iG + 1);  PermG(iG) ← PermG(iG + 1)
end for
%% ——— SUPPRIMER LES TRAMES CORROMPUES ——— %%
if iG > 0 then                                      ▷ nouveau glitchs trouvés
    for i = 1 → iG do                               ▷ corriger l'ordre des canaux
        old_data_r ← data_r
        data_r(1 : PermG(i), TagG(i) : N_r) ← old_data_r(N_c − PermG(i) + 1 : N_c, TagG(i) : N_r)
        data_r(PermG(i) + 1 : N_c, TagG(i) : N_r) ← old_data_r(1 : N_c − PermG(i), TagG(i) : N_r)
    end for
    old_data_r ← data_r
    old_NGindex ← NGindex
    for i = 1 → iG do                               ▷ supprimer les trames corrompues
        data_r(:, TagG(i) − i + 1 : N_r − i) ← old_data_r(:, TagG(i) + 1 : N_r)
        NGindex(TagG(i) − i + 1 : N_r − i) ← old_NGindex(TagG(i) + 1 : N_r)
    end for
    N_r ← N_r − iG
else                                                ▷ pas de nouveaux glitchs trouvés
    RUN ← 0
end if
if ITER >= MITER then
    RUN ← 0
end if
```

Algorithme 5 Algorithme « Prototype » Récursif: Partie TagWeight

```
TagWeight                                                              ——— %%
%% ——— BALISER LES TRAMES CORROMPUES
for n = 1 → N_f do
    valid ← 0
    for k = 1 → N_r do
        if NGindex(k) == n then
            valid ← 1
        end if
    end for
    if valid == 0 then                          ▷ baliser la trame corrompue et ses voisines
        i1 ← max(1, n − DD);   i2 ← min(N_f, n + DD − 1)
        Fvalid(i1 : i2) ← 0
    end if
end for                                                                ——— %%
%% ——— ATTRIBUER DES POIDS POUR L'ALGORITHME VITERBI
for n = 1 → N_f do
    for i = 1 → N_c do
        if Fvalid(n) == 0 then
            Gmetric((n − 1) · N_f + i) ← 1
        else                                    ▷ baliser les échantillons valides
            Gmetric((n − 1) · N_f + i) ← Mweight
        end if
    end for
end for
```

Algorithme 6 Algorithme de Viterbi Pondéré Partie 1: Initialisation

—— Entrée

$CorruptedScan$: $matrix\ [11 \times N_f]$ ▷ balayage corrompu – trames 37 à 485
$Gmetric$: $vector\ [1 \times 11 \cdot N_f]$ ▷ poids de la métrique de branche pour chaque échantillon dans *CorruptedScan*

—— Sortie

$CorrectedScan$: $matrix\ [11 \times N_f]$ ▷ balayage corrigé – trames 37 à 485
$cweight$: ▷ distance cumulée du chemin du balayage corrigé

%% —— INITIALISATION —— %%

$N_c \leftarrow 11$; $N_s \leftarrow 2N_c$ ▷ CONST: #canaux ; #états de l'algorithme de Viterbi
$WD_1 \leftarrow \sqrt{\pi}$ ▷ CONST: multiplicateur de mesure de branche
$FUTUR \leftarrow 10$ ▷ CONST: #échantillons futurs dans la métrique de branche
$d_0(1:N_s) \leftarrow 0$; $d_1(1:N_s) \leftarrow 0$ ▷ distances de branche de treillis
$distances\_cum(1:N_s) \leftarrow \infty$; $distances\_cum(1) \leftarrow 1$ ▷ distance cumulée pour chaque état
$last\_index(1:N_s) \leftarrow N_s$ ▷ indice du dernier échantillon valide reçu
for $i = 1 \rightarrow N_c$ **do** ▷ placer le balayage dans le format vecteur et initialiser les données en sortie
  for $j = 1 \rightarrow N_f$ **do**
    $data((j-1) \cdot N_f + i) \leftarrow CorruptedScan(i,j)$
    $CorrectedScan(i,j) \leftarrow 0$
  end for
end for
$sample\_received(1:(N_f \cdot N_c), 1:N_s) \leftarrow 0$ ▷ initialiser les vecteurs de chemin d'état
for $k = 1 \rightarrow N_s$ **do**
  for $n = 1 \rightarrow N_c$ **do**
    $sample\_received(n,k) \leftarrow scan(n)$
  end for
end for
for $n = 1 \rightarrow N_c \cdot N_f$ **do** ▷ rechercher des poids ∞
  if $Gmetric(n) == \infty$ **then**
    $Prune(n) \leftarrow 1$
  else
    $Prune(n) \leftarrow 0$
  end if
end for
$dd(1:FUTUR) \leftarrow 0$; $pre\_k \leftarrow 0$; $dist_0 \leftarrow 0$; $dist_1 \leftarrow 0$; $sid \leftarrow 0$; $cweight \leftarrow 0$ ▷ variables de travail et mémoires tampons
$old\_sample\_received(1:(N_f \cdot N_c), 1:N_s) \leftarrow 0$
$old\_distances\_cum(1:N_s) \leftarrow 0$
$old\_last\_index(1:N_s) \leftarrow 0$
... suite à l'algorithme 7

Algorithme 7 Algorithme de Viterbi Pondéré Partie 2: Boucle principale

.... suite de l'algorithme 6

%% ——— PRINCIPAL: BOUCLER SUR LES ÉCHANTILLONS ——— %%

for $n = N_c + 1 \rightarrow N_c \cdot N_f - FUTUR$ **do**

  for $k = 1 \rightarrow N_s$ **do** ▷ calculer la métrique de branche de l'échantillon *n* pour les transitions *k* → *k*

    $d_0(k) \leftarrow \text{abs}(data(n) - sample\_received(k, last\_index(k) - N_c + 1))^{\frac{1}{2}}$

  end for

  if $Prune == 0$ **then**

    for $k = 1 \rightarrow N_s$ d ▷ calculer la métrique de branche de l'échantillon *n* pour les transitions *k-1* → *k*

      $dd(1 : FUTUR) \leftarrow \text{abs}(data(n+1 : n+FUTUR) - sample\_received(k, last\_index(k) - N_c + 1))^{\frac{1}{2}}$

      $dd \leftarrow \text{sort}(dd)$ ▷ trier dans l'ordre ascendant

      $d_1(k) \leftarrow 0$

      for $j = 1 \rightarrow FUTUR/2$ **do**

        $d_1(k) \leftarrow d_1(k) + WD_1 \cdot dd(j)/FUTUR$

      end for

    end for

    $old\_distances\_cum \leftarrow distances\_cum$

    $old\_last\_index \leftarrow last\_index$

    $old\_sample\_received \leftarrow sample\_received$ ▷ actualiser les vecteurs d'état

    for $k = 1 \rightarrow N_s$ **do**

      $pre\_k \leftarrow k - 1$

      if $k == 0$ **then**

        $pre\_k \leftarrow N$

      end if

      $dist_0 \leftarrow old\_distances\_cum(k) + d_0(k)$ ▷ distance de branche *k* → *k*

      $dist_1 \leftarrow old\_distances\_cum(pre\_k) + Gmetric(n) \cdot \text{mea}$ ▷ distance de branche *k-1* → *k*

      If $dist_1 < dist_0$ **then** ▷ échantillon considéré comme étant un glitch

        $distances\_cum(k) \leftarrow dist_1$

        $sample\_received(:, k) \leftarrow old\_sample\_received(:, pre\_k)$

        $last\_index(k) \leftarrow old\_last\_index(pre\_k)$

      else ▷ échantillon considéré comme étant valide

        $distances\_cum(k) \leftarrow dist_0$

        $last\_index(k) \leftarrow last\_index(k) + 1$

        $sample\_received(last\_index(k), k) \leftarrow da$

      end if

    end for

  else ▷ les branches *k-1* → *k* n'existent pas

    for $k = 1 \rightarrow N_s$ **do**

      $distances\_cum(k) \leftarrow distances\_cum(k) + d_0(k)$

      $last\_index(k) \leftarrow last\_index(k) + 1$

      $sample\_received(last\_index(k), k) \leftarrow data(n)$

    end for

  end if

end for

%% ——— DECISION ——— %%

$sid = \text{index\_of\_min}(distances\_cum)$ ▷ chemin avec la distance cumulée la plus faible

$cweight = distances\_cum(sid)$ ▷ distance cumulée correspondante

for $i = 1 \rightarrow N_c$ **do**

  for $j = 1 \rightarrow N_f$ **do**

    $CorrectedScan(i, j) \leftarrow sample\_received((j-1) \cdot N_f + i, sid)$ ▷ placer le balayage corrigé dans un format matriciel

  end for

end for

SORTIE $CorrectedScan$; $cweight$

REVENDICATIONS

1. Un procédé de détection et/ou de correction automatique d'erreurs dans un flux de données multiplexées, comprenant :

l'obtention d'un flux de données multiplexées contenant des échantillons de données, ledit flux de données comprenant une pluralité de canaux de données entrelacés organisés de manière cyclique en trames, chaque canal de données possédant un intervalle attribué à l'intérieur de chaque trame, dans lequel intervalle est situé un échantillon de données appartenant vraisemblablement audit canal, lesdits échantillons de données comprenant des échantillons de données ordinaires et des échantillons de données supplémentaires, lesdits échantillons de données supplémentaires ne pouvant pas être attribués à l'un quelconque des canaux entrelacés et provoquant un décalage d'échantillons de données ordinaires subséquents par rapport aux intervalles, et

l'identification desdits échantillons de données supplémentaires par l'exécution d'un algorithme de programmation dynamique, possédant en tant qu'états associés à chaque échantillon de données, des décalages possibles provoqués par des échantillons de données supplémentaires reçus jusqu'à l'échantillon de données associé et dans lequel des transitions autorisées entre états comprennent une première transition, représentant une réception d'un échantillon de données ordinaire et, par conséquent, une constance de décalage, et une deuxième transition, représentant une réception d'un échantillon de données supplémentaire et, par conséquent, un accroissement de décalage d'une unité ; et

dans lequel chaque transition autorisée possède un poids qui lui est associé, le poids associé à une première transition d'un premier état vers un deuxième état dépendant d'une différence entre l'échantillon de données associé audit deuxième état et un échantillon de données prédécesseur direct présumé dans le même canal de données que ledit

échantillon de données associé audit deuxième état, et le poids associé à une deuxième transition d'un premier état vers un deuxième état dépendant d'une différence entre au moins un échantillon de données subséquent audit échantillon de données associé audit deuxième état et un échantillon de données prédécesseur direct présumé dans le même canal de données que ledit au moins un échantillon de données subséquent.

2. Le procédé selon la revendication 1, comprenant la production d'un flux corrigé de données multiplexées qui diffère du flux de données multiplexées en ce qu'il ne contient pas lesdits échantillons de données supplémentaires identifiés.

3. Le procédé selon la revendication 1 ou 2, dans lequel ladite exécution dudit algorithme de programmation dynamique comprend la construction d'une pluralité de chemins au travers desdits états, chacun desdits chemins de ladite pluralité de chemins correspondant à une séquence spécifique de première et deuxième transitions, chacun desdits chemins de ladite pluralité de chemins possédant associé à celui-ci un poids cumulé correspondant à la somme des poids associés aux première et deuxième transitions du chemin, et la recherche d'un chemin de poids minimal parmi ladite pluralité de chemins.

4. Le procédé selon la revendication 3 lorsqu'elle dépend de la revendication 2, dans lequel chacun desdits chemins de ladite pluralité de chemins possède associé à celui-ci une séquence d'échantillons de données présumés ordinaires, et dans lequel ledit flux corrigé de données multiplexées correspond à la séquence d'échantillons de données associée au chemin de poids minimal trouvé.

5. Le procédé selon l'une quelconque des revendications précédentes, dans lequel, avant l'exécution dudit algorithme de programmation dynamique, des trames susceptibles de contenir au moins un échantillon de données supplémentaire sont identifiées.

6. Le procédé selon la revendication 5, dans lequel les poids associés auxdites première et deuxième transitions d'un premier état vers un deuxième état sont modifiées selon que l'échantillon de données associé audit deuxième état appartient ou non à une trame identifiée comme étant susceptible de contenir au moins un échantillon de données supplémentaire.

7. Un programme informatique comprenant des instructions pouvant être mises en œuvre par un processeur, qui, lorsqu'elles sont exécutées par un processeur, amènent ledit processeur à exécuter un procédé selon l'une quelconque des revendications 1 à 6.

8. Un dispositif informatique comprenant un processeur et une mémoire de préférence non volatile contenant en mémoire des instructions pouvant être mises en œuvre par un processeur, qui, lorsqu'elles sont exécutées par ledit processeur, amènent ledit processeur à exécuter un procédé selon l'une quelconque des revendications 1 à 6.

Fig. 1

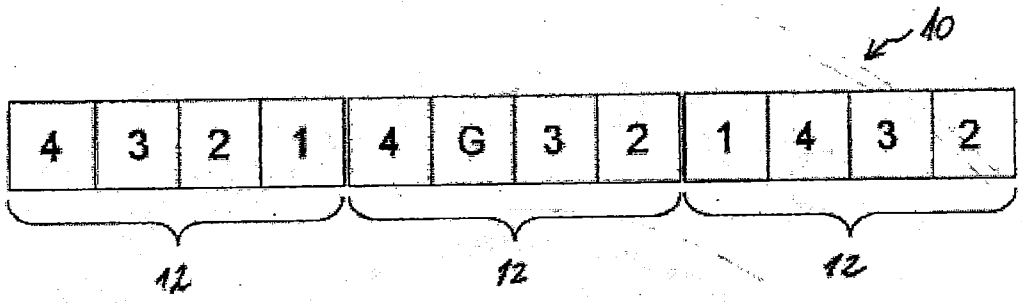

10
4 3 2 1 4 G 3 2 1 4 3 2
12
12
12

Fig. 2

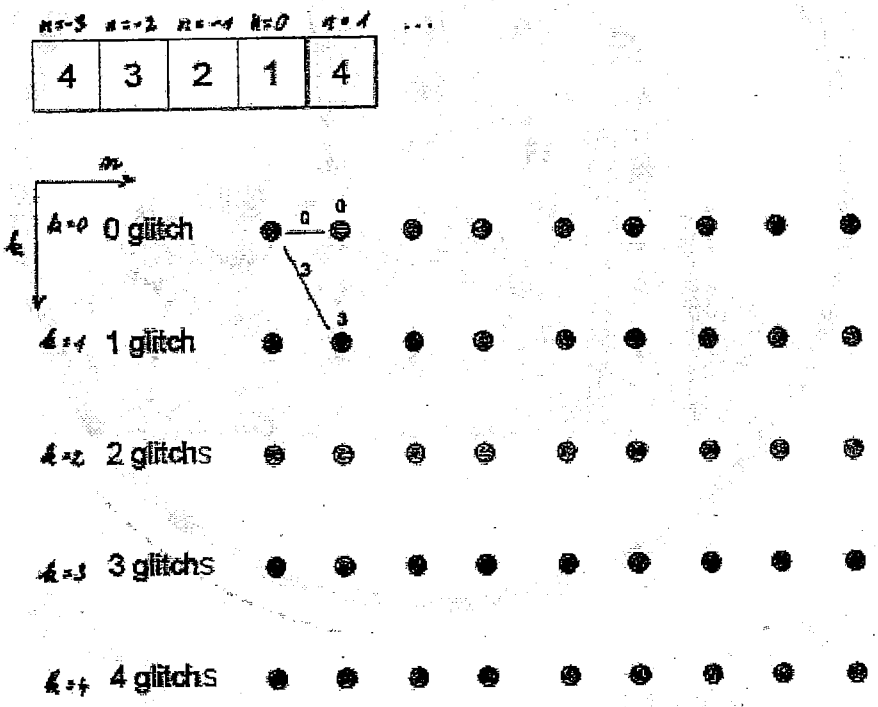

n=-3 n=-2 n=-1 n=0 n=1 ...
4 3 2 1 4
n
k
k=0 0 glitch
0
0
3
3
k=1 1 glitch
k=2 2 glitchs
k=3 3 glitchs
k=4 4 glitchs

Fig. 3

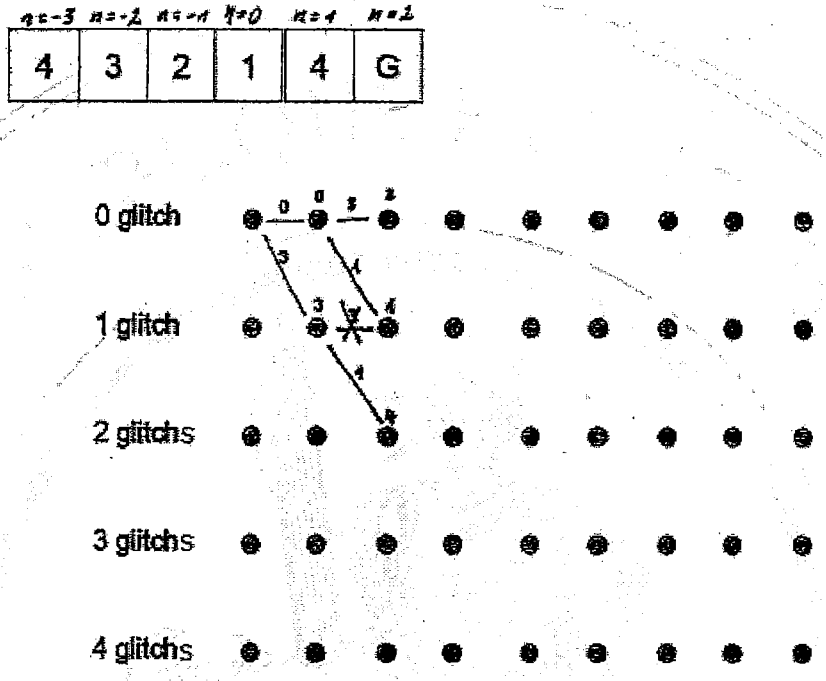

4
3
2
1
4
G
0 glitch
1 glitch
2 glitchs
3 glitchs
4 glitchs

Fig. 4

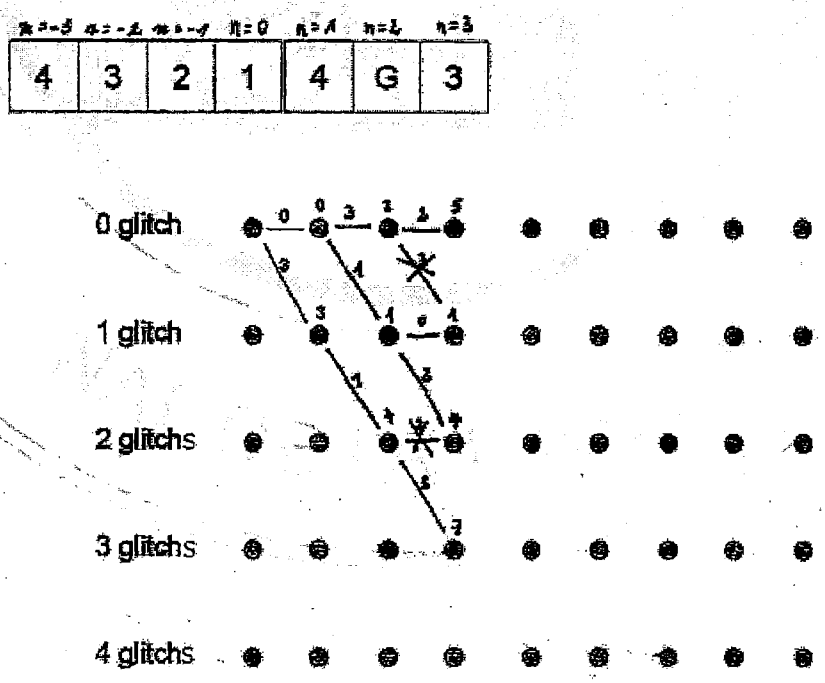
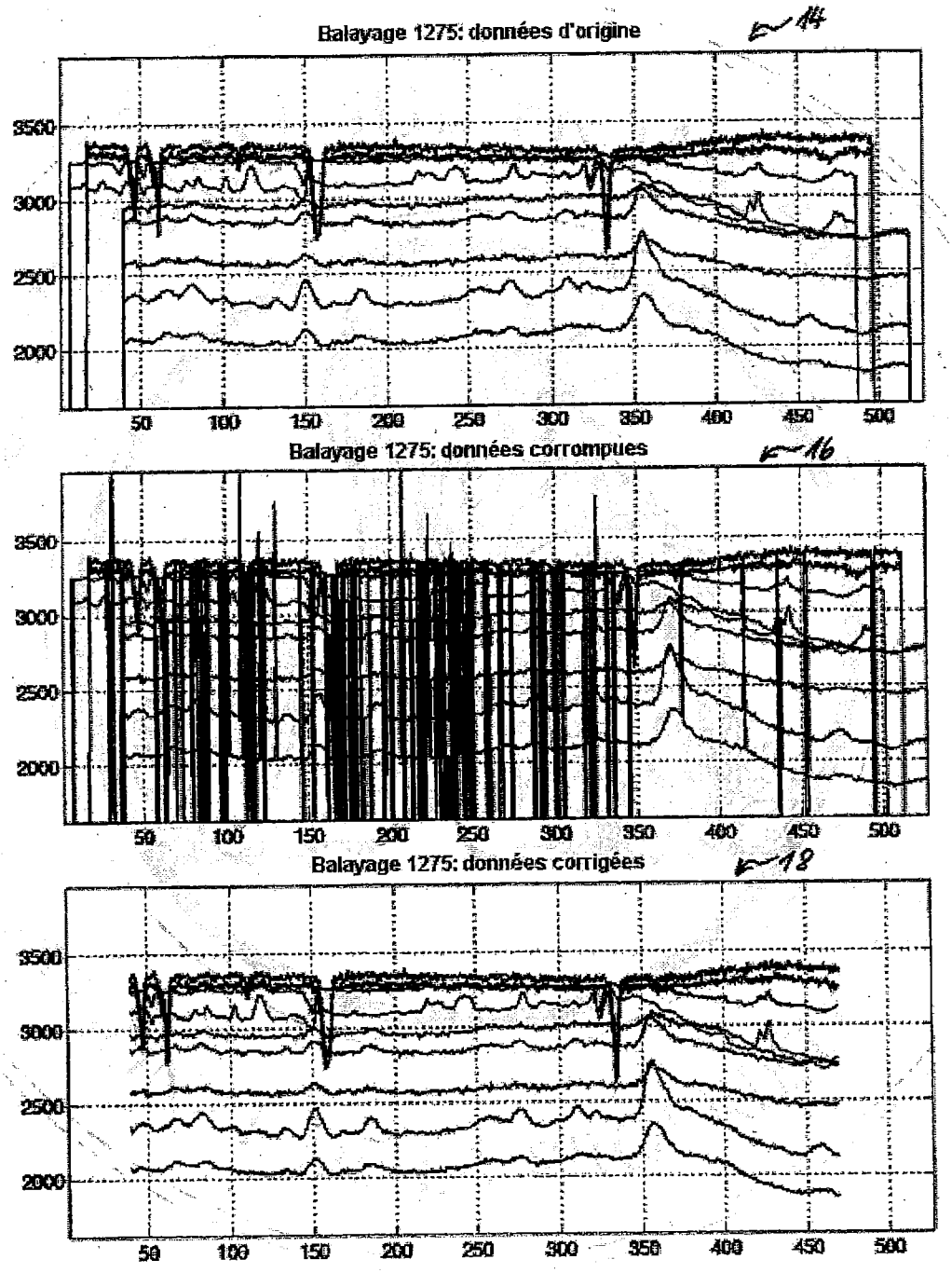

4
3
2
1
4
G
3
0 glitch
1 glitch
2 glitchs
3 glitchs
4 glitchs

Fig. 5


Balayage 1275: données d'origine
14
3500
3000
2500
2000
50
100
150
200
250
300
350
400
450
500
Balayage 1275: données corrompues
16
3500
3000
2500
2000
50
100
150
200
250
300
350
400
450
500
Balayage 1275: données corrigées
18
3500
3000
2500
2000
50
100
150
200
250
300
350
400
450
500

5

N° d'enregistrement national : 1355645

N° de publication : 3007232

RAPPORT DE RECHERCHE

articles L.612-14, L.612-17 et R.612-53 à 69 du code de la propriété intellectuelle

OBJET DU RAPPORT DE RECHERCHE

L'I.N.P.I. annexe à chaque brevet un "RAPPORT DE RECHERCHE" citant les éléments de l'état de la technique qui peuvent être pris en considération pour apprécier la brevetabilité de l'invention, au sens des articles L. 611-11 (nouveauté) et L. 611-14 (activité inventive) du code de la propriété intellectuelle. Ce rapport porte sur les revendications du brevet qui définissent l'objet de l'invention et délimitent l'étendue de la protection.

Après délivrance, l'I.N.P.I. peut, à la requête de toute personne intéressée, formuler un "AVIS DOCUMENTAIRE" sur la base des documents cités dans ce rapport de recherche et de tout autre document que le requérant souhaite voir prendre en considération.

CONDITIONS D'ÉTABLISSEMENT DU PRÉSENT RAPPORT DE RECHERCHE

- [x] Le demandeur a présenté des observations en réponse au rapport de recherche préliminaire.
- [ ] Le demandeur a maintenu les revendications.
- [x] Le demandeur a modifié les revendications.
- [ ] Le demandeur a modifié la description pour en éliminer les éléments qui n'étaient plus en concordance avec les nouvelles revendications.
- [ ] Les tiers ont présenté des observations après publication du rapport de recherche préliminaire.
- [ ] Un rapport de recherche préliminaire complémentaire a été établi.

DOCUMENTS CITÉS DANS LE PRÉSENT RAPPORT DE RECHERCHE

La répartition des documents entre les rubriques 1, 2 et 3 tient compte, le cas échéant, des revendications déposées en dernier lieu et/ou des observations présentées.

- [x] Les documents énumérés à la rubrique 1 ci-après sont susceptibles d'être pris en considération pour apprécier la brevetabilité de l'invention.
- [ ] Les documents énumérés à la rubrique 2 ci-après illustrent l'arrière-plan technologique général.
- [ ] Les documents énumérés à la rubrique 3 ci-après ont été cités en cours de procédure, mais leur pertinence dépend de la validité des priorités revendiquées.
- [ ] Aucun document n'a été cité en cours de procédure.

N° d'enregistrement national : 1355645 N° de publication : 3007232

1. ELEMENTS DE L'ETAT DE LA TECHNIQUE SUSCEPTIBLES D'ETRE PRIS EN CONSIDERATION POUR APPRECIER LA BREVETABILITE DE L'INVENTION

US 4 641 327 A (WEI LEE-FANG [US])
3 février 1987 (1987-02-03)

US 6 023 477 A (DENT PAUL W [US])
8 février 2000 (2000-02-08)

2. ELEMENTS DE L'ETAT DE LA TECHNIQUE ILLUSTRANT L'ARRIERE-PLAN TECHNOLOGIQUE GENERAL

NEANT

3. ELEMENTS DE L'ETAT DE LA TECHNIQUE DONT LA PERTINENCE DEPEND DE LA VALIDITE DES PRIORITES

NEANT