(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 799 027**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **00 10125**

(51) Int Cl[7] : **G 06 T 5/00,** G 06 F 5/50

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 01.08.00.

(30) **Priorité :** 02.08.99 US 09365583.

(43) **Date de mise à la disposition du public de la demande :** 30.03.01 Bulletin 01/13.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Ce dernier n'a pas été établi à la date de publication de la demande.*

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

(71) **Demandeur(s) :** *SCHLUMBERGER TECHNOLOGIES INC* — US.

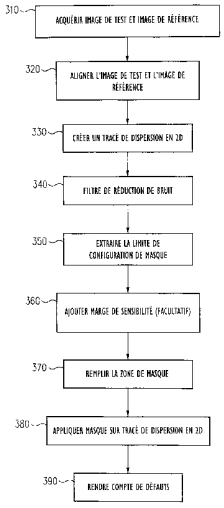
(72) **Inventeur(s) :** AGHAJAN HAMID K.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** REGIMBEAU.

(54) PROCEDE DE DETECTION DE DEFAUTS ET MOYENS DE MEMOIRE POUR LA MISE EN OEUVRE DE CELUI-CI.

(57) La présente invention concerne un procédé de détection de défauts. Il comprend les étapes consistant à: (a) acquérir (310) une première image d'un objet à inspecter et une deuxième image associée; (b) aligner (320) la première image avec la deuxième image; (c) créer (330) un premier tracé en traçant les niveaux de gris des pixels provenant de la première image en fonction des niveaux de gris des pixels correspondants provenant de la deuxième image; (d) créer (340) un deuxième tracé en filtrant le premier tracé; (e) créer (350) un masque tel que le profil du masque est défini par la configuration du deuxième tracé; et (f) utiliser (380) le masque pour détecter des défauts représentés dans la première image. Un filtrage (340) peut être exécuté en utilisant un filtre morphologique. Une extension du masque peut être ajustée (360) par l'utilisateur. L'invention concerne en outre un support lisible par informatique utilisé pour la mise en oeuvre de ce procédé.



FR 2 799 027 - A1

Une fraction de l'exposé de ce document de brevet contient des éléments qui sont sujets à protection par la loi sur les droits d'auteur. Le propriétaire des droits d'auteur n'élève aucune objection à ce que quiconque reproduise par fac-similé le document de brevet ou l'exposé du brevet, tel qu'il apparaît dans les collections ou les archives de brevets du Patent and Trademark Office, mais se réserve par ailleurs tous droits d'auteurs, quels qu'ils soient.

La présente invention concerne de façon générale un traitement d'image numérique et, plus particulièrement, des systèmes et des procédés qui utilisent des techniques de comparaison d'images pour détecter des défauts dans un dispositif semi-conducteur.

Des techniques de comparaison d'images sont utilisées pour détecter des défauts dans une tranche semi-conductrice. Typiquement, une image de test est acquise et est ensuite comparée à une image de référence. Un algorithme de détection de défauts est ensuite utilisé pour détecter des différences ou, en d'autres termes, des variations entre les images et pour déterminer si ces variations sont des défauts réels. Dans le mode d'inspection dit à logique aléatoire, une image d'une première microplaquette est acquise et est ensuite comparée à l'image d'une deuxième microplaquette de la même tranche. Un mode d'inspection par ensembles est effectué d'une façon semblable, sauf qu'une section d'une microplaquette est comparée à une autre section, à structure identique, de la même microplaquette. Un mode d'inspection par ensembles est utilisé, par exemple, pour tester des dispositifs à structures répétitives, par exemple des cellules de mémoires. Au lieu de comparer des images provenant d'une tranche en cours de test, des défauts peuvent aussi être détectés en comparant l'image de test acquise à une image bonne connue, provenant d'une base de données.

La Figure 1 illustre un procédé de détection de défauts de l'art antérieur.

À l'étape 110, une image de test et une image de référence de la particularité de la tranche en cours d'analyse sont acquises à partir de sections différentes de la tranche en utilisant, par exemple, des techniques

classiques d'imagerie par faisceau électronique. Chaque image comprend une série de pixels et chaque pixel est défini par son emplacement à l'intérieur de l'image et son intensité et son niveau de gris. L'utilisation de niveaux de gris en traitement d'images est connue dans l'art et décrite dans l'ouvrage "Digital Image Processing", c'est-à-dire Traitement d'image numérique, de R. C. Gonzales et R. E. Woods, Addison-Wesley (1992) incorporé ici par référence en totalité, par exemple pages 6 et 7.

À l'étape 120, les deux images sont ensuite alignées pixel par pixel de façon que chaque particularité de l'image de test concorde avec la particularité correspondante de l'image de référence.

À l'étape 130, une image de différence est ensuite engendrée en soustrayant les niveaux des gris des deux images. Puisque ce sont des pixels concordants à niveaux de gris identiques qui sont soustraits, l'image de différence représente des variations du niveau de gris des pixels entre l'image de référence et l'image de test.

À l'étape 140, le niveau de gris de chaque pixel contenu dans l'image de différence est mis à l'échelle, normalisé, et ensuite tracé pour former un histogramme unidimensionnel comme l'histogramme 200 de la Figure 2. L'histogramme 200 est un tracé qui représente le nombre de pixels, contenus dans l'image de différence, qui possèdent un niveau de gris spécifique. Par exemple, l'histogramme 200 indique qu'il existe, dans l'image de différence, 20.000 pixels dont le niveau de gris est de 50.

Un pixel provenant de l'image de test peut être différent d'un pixel correspondant de l'image de référence, même en l'absence de défauts dans les deux images. Des variations d'intensité peuvent par exemple être provoquées par des différences des structures de couches physiques, par un bruit dans l'électronique d'acquisition d'image et dans les trajets de signaux, et par un niveau variable de modulation de bruit à l'intérieur d'une image unique pour des niveaux de bruits différents. Par conséquent, des pixels présents dans l'image de différence n'indiquent pas nécessairement qu'il existe un défaut.

À l'étape 150 de la Figure 1, chaque pixel de l'image de différence est donc comparé à une fenêtre de seuil pour différencier entre des défauts

réels et de faux défauts, appelés aussi "nuisances". Des pixels dont le niveau de gris est à l'extérieur de la fenêtre de seuil sont déclarés comme défauts.

À l'étape 160 de la Figure 1, si le niveau de seuil est par exemple de ± 50, un événement de défaut est déclaré si le niveau de gris d'un pixel de l'image de différence est de 60, c'est-à-dire si les niveaux de gris des images de test de référence diffèrent de 60 unités. Pour assurer que la microplaquette est réellement défectueuse, l'événement de défaut est ensuite vérifié par un opérateur avant un rebut éventuel de la microplaquette dans un traitement ultérieur.

Trouver la valeur optimale de seuil pour une image de données de test est une tâche importante mais imprécise. La valeur de seuil doit être choisie de façon que des défauts réels soient détectés, tout en les différenciant des défauts de nuisance. Plus la valeur de seuil est étroite, plus grand est le nombre de défauts de nuisance déclarés. Des défauts de nuisance affectent de façon défavorable le débit de production parce que chaque événement de défaut doit être contrôlé et vérifié. En revanche, élargir la fenêtre de seuil réduit les événements de défaut de nuisance aux dépens d'une absence de détection de défauts réels.

C'est donc le but de la présente invention que de fournir un procédé de détection de défauts, et un support lisible par informatique, qui remédient aux difficultés exposées ci-dessus et permettent de détecter des défauts réels tout en minimisant le nombre de détections défauts de nuisance.

Ce but est atteint, selon un premier aspect de l'invention, par un procédé de détection de défauts caractérisé en ce qu'il comprend les étapes consistant à :

(a) acquérir une première image d'un objet à inspecter et une deuxième image associée ;

(b) aligner la première image avec la deuxième image ;

(c) créer un premier tracé en traçant les niveaux de gris des pixels provenant de la première image en fonction des niveaux de gris des pixels correspondants provenant de la deuxième image ;

(d) créer un deuxième tracé en filtrant le premier tracé ;

(e) créer un masque tel que le profil du masque est défini par la configuration du deuxième tracé ; et

(f) utiliser le masque pour détecter des défauts représentés dans la première image.

Un filtrage peut être exécuté en utilisant un filtre morphologique.

On peut prévoir qu'une extension du masque puisse être ajustée par l'utilisateur.

Le procédé peut comprendre en outre l'utilisation d'un filtre à moyenne mobile pour lisser le profil du masque.

La deuxième image peut être obtenue à partir d'une base de données.

Le procédé peut comprendre en outre une mémorisation des tracés résultant des étapes (c) et (d) dans un support lisible par informatique.

Selon un deuxième aspect, l'invention réalise un support lisible par informatique caractérisé en ce qu'il mémorise un programme de mise en œuvre du procédé exposé ci-dessus.

Selon un troisième aspect, l'invention réalise un support lisible par informatique, caractérisé en ce qu'il comprend :

— une série d'emplacements de mémoire qui contiennent des données représentant une première image et une deuxième image qui lui est associée, lesdites première et deuxième image incluant chacune une série de pixels qui sont définis chacun par une coordonnée d'emplacement et par un niveau de gris ; et

— un ensemble qui comprend une série d'emplacements de mémoire mémorisant des données définissant un masque, le masque étant créé en filtrant un tracé des niveaux de gris des pixels provenant de la première image en fonction des niveaux de gris de pixels correspondants provenant de la deuxième image.

Le filtrage peut être exécuté en utilisant un filtre morphologique.

Un algorithme à moyenne mobile peut être utilisé pour lisser le tracé des niveaux de gris de pixels provenant de la première image en fonction des niveaux de gris de pixels provenant de la deuxième image.

Les buts, particularités et avantages de la présente invention exposés ci-dessus ainsi que d'autres ressortiront davantage de la description qui suit de modes de réalisation préférés en conjonction avec les dessins dans lesquels :

- la Figure 1 représente un schéma logique d'un procédé de détection de défauts de l'art antérieur ;
- la Figure 2 représente un tracé d'histogramme unidimensionnel de niveaux de gris ;
- la Figure 3 représente sous forme de schéma logique les étapes d'un mode de réalisation de l'invention ;
- les Figures 4 A à 4C représentent une étape d'alignement selon la présente invention ;
- la Figure 5 représente un tracé bidimensionnel de dispersion conforme à la présente invention ;
- les Figures 6 et 7 représentent respectivement une image de test et une image de référence, prises sur une tranche de dispositif ;
- la Figure 8 représente un tracé bidimensionnel de dispersion conforme à la présente invention ;
- les Figures 9A et 9B représentent le résultat de l'application d'un filtre morphologique sur le tracé bidimensionnel de dispersion de la Figure 8 ;
- les Figures 10A à 10C représentent des profils de distances unidimensionnels selon la présente invention ;
- la Figure 11 représente un masque conforme à la présente invention ;
- la Figure 12 représente un masque superposé à un tracé bidimensionnel de dispersion non filtré ;
- la Figure 13 représente une carte de défauts de tranche obtenue en utilisant un masque adaptatif ;
- la Figure 14 représente un seuil prédéterminé superposé à un tracé bidimensionnel de dispersion non filtré ; et
- la Figure 15 représente une carte de défauts de tranche obtenue en utilisant un seuil prédéterminé.

La présente invention remédie aux limitations des procédés de détection de défauts de l'art antérieur en utilisant une méthode à seuil adaptatif appliquée à une paire d'images en cours d'analyse. À la différence des procédés de l'art antérieur qui utilisent un seuil prédéterminé pour toutes les paires d'images, ce procédé utilise un masque de seuil qui est adapté à chaque paire d'images. L'invention peut être utilisée dans diverses applications d'imagerie, y compris des systèmes d'inspection à faisceaux électroniques, à fond clair, à fond noir, par laser et par microscopie à force atomique, ou "AFM" selon les initiales du terme anglo-saxon du terme atomic-force microscopy.

La Figure 3 représente sous forme de schéma logique les étapes d'un mode de réalisation conforme à la présente invention.

À l'étape 310, une image de test et une image de référence, d'une structure semi-conductrice par exemple, sont acquises en utilisant des techniques classiques d'acquisition d'images. Les images peuvent aussi être acquises en utilisant un système d'acquisition par imagerie pas à pas, ou step-and-image selon le terme anglo-saxon.

À l'étape 320, les images de test et de référence sont alignées pour mettre en concordance des pixels correspondants des deux images. Diverses techniques d'alignement peuvent être utilisées dans le cadre de la présente invention. L'étape d'alignement est nécessaire pour assurer que chaque particularité de l'image de test est comparée à une particularité équivalente de l'image de référence.

L'étape 320 est en outre illustrée aux Figures 4A à 4C. La Figure 4A représente une image de test 410 qui comprend des pixels 411 à 416. Chaque pixel est défini par son niveau de gris et par son emplacement sur l'image. Par exemple, le pixel 413 est à l'emplacement i = 10 et j = 30, appelé emplacement (10, 30). Le niveau de gris du pixel 413 est ici de 50 à titre d'illustration. La table 1 fournit les coordonnées de l'emplacement et le niveau de gris, pour chaque pixel de l'image de test 410, tandis que la table 2 fournit la même information pour les pixels 421 à 426 de l'image de référence 420 de la Figure 4B.

TABLEAU 1

Pixel	Emplacement (i,j)	Niveau de gris
411	(10, 10)	100
412	(10, 20)	150
413	(10, 30)	50
414	(20, 30)	180
415	(20, 20)	200
416	(20, 10)	250

TABLEAU 2

Pixel	Emplacement (i,j)	Niveau de gris
421	(10, 10)	100
422	(10, 20)	150
423	(10, 30)	50
424	(20, 30)	150
425	(20, 20)	100
426	(20, 10)	0

La Figure 4C illustre graphiquement l'alignement de l'image de test 410 et de l'image de référence 420. L'emplacement 431 de pixels alignés comprend les pixels 411 et 421, l'emplacement 432 de pixels alignés comprend les pixels 412 et 422, etc.

Dès lors que les images de test et de référence ont été alignées, la correspondance de pixel à pixel entre l'image de test et l'image de référence est connue.

À l'étape 330 de la Figure 3, un tracé bidimensionnel, appelé aussi "2D", de dispersion est créé en traçant, pour chaque emplacement de pixels alignés, le niveau de gris d'un pixel provenant de l'image de test en fonction du niveau de gris du pixel correspondant de l'image de référence. En utilisant comme exemple la Figure 4C, le niveau de gris du pixel 411 est tracé en fonction du niveau de gris du pixel 421, le niveau de gris du pixel 412 est tracé en fonction du niveau de gris du pixel 422, et ainsi de suite.

Mettre en œuvre l'étape 330 pour les emplacements 431 à 436 fournit les données représentées au Tableau 3. Le tracé de dispersion bidimensionnel résultant 500 est représenté à la Figure 5.

TABLEAU 3

Emplacement de pixels alignés	Niveau de gris d'image de test	Niveau de gris d'image de référence	Coordonnées (t_{gris}, r_{gris})
431	100	100	(100, 100)
432	150	150	(150, 150)
433	50	50	(50, 50)
434	180	150	(180, 150)
435	200	100	(200, 100)
436	250	0	(250, 0)

où t_{gris} et r_{gris} représentent les niveaux de gris de l'image de test et de l'image de référence, respectivement.

Le tableau 3 montre que les niveaux de gris des emplacements 434, 435 et 436 de pixels alignés sont différents, ce qui indique donc la présence de défauts possibles. Les emplacements 431, 432 et 433 sont exempts de défauts parce que les niveaux de gris de l'image de test et de l'image de référence sont les mêmes auxdits emplacements. Le tracé de dispersion 500 de la Figure 5 fournit une information quant à la présence de défauts possibles.

Tous les emplacements de pixels alignés dont les niveaux de gris sont les mêmes peuvent être représentés dans le tracé de dispersion 500 par une ligne imaginaire 501. La pente de la ligne imaginaire 501 est +1 parce qu'elle représente les emplacements des pixels alignés pour lesquels le niveau de gris du pixel de l'image de test est le même que le niveau de gris du pixel correspondant de l'image de référence.

Tous les emplacements de pixels alignés pour lesquels les niveaux de gris sont différents sont éloignés de la ligne imaginaire 501. Plus un emplacement est éloigné de la ligne 501, plus l'écart entre les niveaux de

gris est grand, et plus le risque d'existence d'un défaut à cet emplacement est élevé. Dans le tracé de dispersion 500, les emplacements 434, 435 et 436 ne sont pas sur la ligne imaginaire 501 et indiquent donc la présence de défauts possibles. Dans le présent exposé, l'expression (t_{gris}, r_{gris}), sera utilisée pour désigner les coordonnées d'un point de données du tracé bidimensionnel de dispersion, afin de les distinguer d'un emplacement du pixel d'image, dont les coordonnées sont désignées par l'expression (i, j). Par exemple, l'emplacement 435 de pixels alignés est défini par un point de données du tracé de dispersion bidimensionnel à l'emplacement (200, 100).

Un pseudo code de mise en application d'un tracé bidimensionnel de dispersion par un logiciel informatique ou, en d'autres termes, un logiciel d'ordinateur est représenté ci-dessous. Dans le pseudo code, les valeurs des niveaux de gris sont tracées dans une variable d'ensemble de mémoire ("Dispersion").

```
/* PSEUDO CODE DE CRÉATION D'UN TRACÉ BIDIMENSIONNEL DE DISPERSION */
Acquérir Image de Référence ;
Acquérir Image de Test ;
Aligner Image de Test par rapport à Image de Référence ;
Créer une Image 256x256 appelée Dispersion ;
Initialiser Dispersion à 0 ;
Faire de i = 1 à NumRows
{
        Faire de j = 1 à NumCols
        {
                p1 = Reference (i,j) ;
                p2 = Test(i,j) ;
                Scatter(p2,p1) = 1 ;
        }
}
Tracer Dispersion sous forme d'image ;
/* FIN DU PSEUDO CODE */
```

Les Figures 6 à 8 résument sous forme d'images les étapes 310, 320 et 330 du mode de réalisation représenté à la Figure 3. La Figure 6 représente une image de test 6 acquise classiquement à partir d'une tranche qui inclut un défaut 601. Une image de référence 700 représentée à la Figure 7 est acquise et est ensuite alignée avec l'image de test 600, cet alignement n'étant pas représenté. Le tracé de dispersion bidimensionnel 800 de la Figure 8 est créé en traçant les niveaux de gris de pixels provenant de l'image de test en fonction des niveaux de gris de pixels correspondants provenant de l'image de référence. Le tracé de dispersion peut être engendré à la main ou en utilisant un ordinateur programmé.

Les points de données du tracé de dispersion 800 sont tracés sous forme de points blancs sur un arrière-plan noir. Une ligne 801 définit les emplacements de pixels alignés où les niveaux de gris des pixels des images de test et de référence sont identiques. Par exemple, si l'image de test 600 était identique à l'image de référence 700, tous les points de données des tracés de dispersion 800 sont sur la ligne 801.

Le tracé de dispersion 800 contient une information de niveaux de gris pour tous les pixels des images de test et de référence, y compris les pixels du défaut 601. Comme exposé précédemment, plus un point de données est éloigné de la ligne 801, plus il est probable que ce point de données indique la présence d'un défaut. Le présent procédé tire profit de cette information et construit un "masque" qui peut être "superposé" à un tracé de dispersion 800 afin de différencier entre des pixels qui impliquent des défauts et des pixels indiquant une bonne qualité. Des points de données situés à l'extérieur du masque sont déclarés comme événements de défauts.

Comme indiqué à l'étape 340 de la Figure 3, un filtre de réduction du bruit est appliqué sur les points de données du tracé de dispersion 800 afin de trouver le profil ou les limites du masque. Divers filtres classiques de réduction de bruit peuvent être utilisés dans le cas de la présente invention, y compris par exemple des filtres morphologiques. Des filtres morphologiques sont connus dans l'art et sont décrits dans le document "Digital Image Processing Concepts, Algorithms, and Scientific Applications", c'est-à-dire Concepts, algorithmes et applications scientifiques du traitement

d'image numérique, de B. Jahne, Springer Verlag (1991), Chapitre 11, et dans l'ouvrage "Ditigal Image Processing", c'est-à-dire Traitement d'image numérique, de R. C. Gonzales et R. E. Woods, ADDISON-WESLEY (1992), Chapitre 8, qui sont l'un et l'autre incorporés ici par référence dans leur totalité. Un filtrage morphologique effectue un "compactage" et un "nettoyage" des points de données du tracé de dispersion 800 afin de définir une configuration de masque. Le tracé bidimensionnel de dispersion 950, représenté à la Figure 9A, est le résultat de l'application d'un filtre morphologique sur le tracé de dispersion 800. Le tracé 950 de dispersion contient la configuration de masque 900.

L'étape 350 de la Figure 3 consiste en une extraction de limites, c'est-à-dire le processus d'obtention des coordonnées de chaque point de données limite d'une configuration de masque. Un algorithme d'extraction de la limite de la configuration de masque 900 est le suivant :

ALGORITHME D'EXTRACTION D'UNE CONFIGURATION DE MASQUE

(a1) Comme représenté à la Figure 9B, créer une ligne 901 qui s'étend du coin supérieur gauche au coin inférieur droit du tracé de dispersion 950.

(a2) Créer deux ensembles de nombres pour garder un suivi de distances perpendiculaires entre la ligne 901 et un point de limite de données. Appeler SUPÉRIEUR l'un des ensembles. L'ensemble SUPÉRIEUR est utilisé pour garder un suivi de distances perpendiculaires de points limites situés au-dessus de la ligne 901, c'est-à-dire dans la région désignée par la flèche 902. L'autre ensemble, appelé INFÉRIEUR, est utilisé pour garder un suivi de distances perpendiculaires de points limites situés au-dessous de la ligne 901 c'est-à-dire dans la région désignée par la flèche 903. Un exemple d'une distance perpendiculaire est la longueur de la ligne perpendiculaire 904 qui s'étend de la ligne 901 jusqu'à un point limite 905. Un autre exemple est la longueur de la ligne 906 qui est une ligne perpendiculaire qui s'étend de la ligne 901 au point limite 907.

(a3) Initialiser à un 0 logique tous les éléments des ensembles SUPÉRIEUR et INFÉRIEUR.

(a4) Pour chaque emplacement de coordonnées (t_{gris}, r_{gris}) du tracé de dispersion 950, contrôler s'il existe un point de données pour les coordonnées. Si tel est le cas, poursuivre par les étapes (a5) à (a9) ; dans le cas contraire, passer à l'emplacement suivant du tracé de dispersion. Aux Figures 8, 9A et 9B, tous les points de données sont des points blancs tracés sur un arrière-plan noir, c'est-à-dire qu'un point de données ou un 1 logique est tracé comme point blanc tandis qu'un 0 logique ou une absence de point de données est tracé comme point noir. Par conséquent, les sections sombres de la Figure 9B ne contiennent aucun point de données et sont ignorées.

(a5) S'il existe un point de données pour l'emplacement du tracé de dispersion, mesurer la distance perpendiculaire D_{perp} à la ligne 901. Calculer aussi l'emplacement R_{profil} de ce point de données le long du profil unidimensionnel ou, en d'autres termes, "1D" de distances. Des profils unidimensionnels de distances sont décrits plus loin. R_{profil} peut être calculé en utilisant l'Éq. 1.

$$(\text{Éq. 1})\ R_{profil} = (t_{gris} + r_{gris}) / 2$$

(a6) Si les coordonnées (t_{gris}, r_{gris}) sont au-dessus de la ligne 901, une valeur positive est assignée à D_{perp}. Dans le cas contraire, D_{perp} est négative.

(a7) Si D_{perp} est supérieure à la distance perpendiculaire actuellement mémorisée dans l'élément R_{profil} de l'ensemble SUPÉRIEUR, mémoriser D_{perp} dans l'élément R_{profil} de l'ensemble SUPÉRIEUR.

(a8) Si D_{perp} est inférieure à la distance perpendiculaire actuellement mémorisée dans l'élément R_{profil} de l'ensemble INFÉRIEUR, mémoriser D_{perp} dans l'élément R_{profil} de l'ensemble INFÉRIEUR.

(a9) Continuer pour tous les points de données.

Après avoir exécuté l'algorithme d'extraction de la configuration de masque exposé ci-dessus, les ensembles SUPÉRIEUR et INFÉRIEUR contiennent les distances perpendiculaires des points limites de la configuration de masque.

Il est possible d'utiliser les distances perpendiculaires et leurs R_{profil} correspondants peuvent être utilisés pour créer un profil unidimensionnel 1000 de distances représenté à la Figure 10A. La courbe 1010 est le graphe

des distances perpendiculaires mémorisées dans des éléments R_{profil} de l'ensemble SUPÉRIEUR tandis que la courbe 1020 est un graphe similaire pour l'ensemble INFÉRIEUR. Pour continuer la délinéation de la configuration extraite du masque, le profil 1000 de distance peut être lissé en utilisant par exemple un algorithme à moyenne mobile. Des algorithmes à moyenne mobile sont connus dans l'art et sont décrits dans l'ouvrage "Discrete-Time Signal processing", c'est-à-dire Traitement de signaux discrets dans le temps, de A. V. Oppenheim et R. W. Schafer, Prentice-Hall (1989), incorporé ici par référence dans sa totalité. Le profil 2000 de distances représenté à la Figure 10B est le résultat de l'utilisation d'un algorithme à moyenne mobile sur le profil 1000 de distances. Les courbes 1030 et 1040 sont les moyennes mobiles des courbes 1010 et 1020, respectivement.

Une image de sensibilité peut facultativement être appliquée sur la configuration extraite du masque, comme indiqué à l'étape 360 de la Figure 3, pour permettre à un utilisateur de modifier l'extension du masque. La valeur de sensibilité sélectionnée par l'utilisateur peut être utilisée pour mettre à l'échelle ou déporter la configuration extraite de masque. La courbe 1050 de la Figure 10C représente le résultat de l'addition d'une valeur de sensibilité, S_{valeur}, à chaque point de la courbe 1030. La courbe 1060 est un résultat obtenu en soustrayant, de chaque point de la courbe 1040, S_{valeur} qui est la même valeur de sensibilité.

Une table à consulter, appelée simplement table dans ce qui suit, de masque, est créée en remplissant, comme indiqué à l'étape 370 de la Figure 3, tous les emplacements de coordonnées situés à l'intérieur de la limite de la configuration extraite de masque. Un algorithme de remplissage de la configuration extraite de masque est illustré en utilisant le tracé de dispersion 950 représenté à la Figure 9B.

ALGORITHME DE REMPLISSAGE D'UNE ZONE DE CONFIGURATION DE MASQUE

(b1) Créer un tracé bidimensionnel de dispersion, $M_{dispersion}$. Mettre à la valeur 1 logique tous les points de données de $M_{dispersion}$.

(b2) Pour chaque emplacement (t_{gris}, r_{gris}) du tracé de dispersion 950, calculer R_{profil} en utilisant l'Éq. 1 et obtenir les distances perpendiculaires D_{perp}.

(b3) Tracer R_{profil} et D_{perp} dans le profil 2000 de distances représenté à la Figure 10B, ou dans le profil de distances représenté à la Figure 10C si une valeur ou, en d'autres termes, une marge de sensibilité est utilisée. Si le point (R_{profil}, D_{perp}) est inclus à l'intérieur des courbes 1030 et 1040, restaurer à la valeur 0 logique l'emplacement (t_{gris}, r_{gris}) de $M_{dispersion}$. Dans le cas contraire, continuer à l'emplacement suivant (t_{gris}, r_{gris}) du tracé de dispersion 950.

(b4) Continuer pour tous les emplacements.

Il résulte de l'algorithme précédent un graphe de $M_{dispersion}$ 1100 représentée à la Figure 11. Le tracé $M_{dispersion}$ 1100 contient un masque 1110 qui peut être utilisé pour détecter des points de défauts dans un tracé bidimensionnel de dispersion. Tous les points intérieurs au masque 1110 sont à une valeur "0" logique. La Figure 12 représente le masque 1110 superposé au tracé de dispersion 800. Des points de données extérieurs au masque sont déclarés comme événements de défauts.

Un algorithme d'utilisation du masque 1110 pour détecter des défauts est le suivant :

ALGORITHME DE DÉTECTION DE DÉFAUTS EN UTILISANT UN MASQUE

(c1) Pour tous les pixels d'une image de test et d'une image de référence, lire les niveaux correspondants de gris t_{gris} et r_{gris}, respectivement.

(c2) Si l'emplacement (t_{gris}, r_{gris}) du tracé $M_{dispersion}$ 1100 est un 0 logique, ceci indique que l'emplacement est intérieur au masque et qu'il n'existe donc aucun événement de défaut. Continuer au pixel suivant des images de test et de référence.

(c3) Si l'emplacement (t_{gris} , r_{gris}) du tracé $M_{dispersion}$ 1100 est un 1 logique, l'emplacement est extérieur au masque et il existe un défaut. Rendre compte d'un événement de défaut.

(c4) Continuer pour toutes les paires de pixels et les images de test et de référence.

En dernière position dans la description, on fournit des exemples additionnels de modes de mise en application de la présente invention. On liste dans cette dernière partie le code source d'une fonction de langage de programmation "C" selon la présente invention. Le code serait exécuté par un ordinateur ou un processeur qui est couplé classiquement à un système d'inspection de défauts ou qui en fait partie. Évidemment, un tel système mémoriserait typiquement ce code source et les tracés, masques, etc., résultants, dans un support lisible par informatique, qui est une mémoire. La Table 4 représente la correspondance entre les étapes de l'invention et le code source listé dans l'Annexe A.

TABLE 4

Fonction C	Page d'Ann A	Étape	Commentaire
hist2D8	A/3	330	Tracé 2D de dispersion
hist2D8 open	A/3	340	Filtre Morphologique
hist_2D8_1Dprofile	A/4	350	Extraits profil 1D et appliquer une moyenne mobile
hist_2D8_fitbound	A/5	360, 370	Marge de sensibilité et remplissage de masque
hist_2D8_thresh	A/6	380	Contrôler quant à défauts (effet de seuil)

Les Figures 13 à 15 démontrent de façon plus détaillée l'efficacité du présent procédé. Utiliser le tracé $M_{dispersion}$ 1100 pour détecter des défauts sur le tracé de dispersion 800 en utilisant les étapes (c1) à (c4) fournit une carte 1300 de défauts représentée à la Figure 13. On notera que la carte 1300 de défauts identifie correctement le défaut 601 de l'image de test 600 de la Figure 6.

La Figure 14 représente sous forme de graphique l'application, sur le tracé de dispersion 800, d'un seuil prédéterminé, défini par des lignes 1401 et 1402. Des points qui ne sont pas entre les lignes 1401 et 1402 sont déclarés comme événements de défauts. La Figure 15 représente une carte de défauts qui résulte d'une application du seuil prédéterminé, sur le tracé de dispersion 800. On notera que de nombreux défauts de nuisance ont été détectés alors que le défaut 601 ne l'a pas été.

Il faut comprendre que la description donnée ci-dessus ne l'est qu'à titre d'illustration et ne doit pas être comprise comme limitative. De nombreuses variantes sont possibles sans s'écarter de l'esprit et du cadre de l'invention. La présente invention est exposée par les revendications qui suivent.

```
/*************************************************************
   Returns:
   0 for success
   -1 for functional failure
   -2 if algorithm fails to find acceptable threshold
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <mp.h>
#include <image.h>
#include <SLBapi.h>
#include <pgm.h>
#include <morph1.h>
#include <scatter2d8.h>

#define SQRT2              1.414213562373          /*sqrt(2)*/
#define MIN_DIST           1                       /*min image difference used as the
                           beginning of 1D profiles*/
#define MVA_SIZE           21                      /*size of moving average window*/
#define MVA_RATIO          0.9                     /*allowable shrinkage in 1d profile
                           wrt its mva*/
#define MIN_DATA_SIZE  80                          /*min # of data points to use curve
                           fitting*/
#define MX_SIZE            (3 << GRAY_SCALE_BITS)  /*size of the data matrix for   curve
                           fitting*/
#define HIST_MORPH_PASSES 1                        /*no. of passes for open/close
                           operations*/
#define SIGMA_MAX          5                       /*enable removal of outliers if sigma
                           of fit is bigger than this*/
/*#define MIN_THR          10                      /*min thr below which no difference
                           is called a defect*/
#define EPSILON            0.1                     /*min value for det(AtA)*/
#define ALG_FITMASK        1                       /*detection alg using cleaned 2D hist
                           as a mask*/


static void hist2D8(unsigned char *a, unsigned long tcols_a,
                    unsigned char *b, unsigned long tcols_b,
                    unsigned long ncols, unsigned long nrows,
                    unsigned char *table);

static void hist2D8_open(unsigned char *table,
                         unsigned char *hist2DBitMap, long Passes);

static void hist2D8_1Dprofile(unsigned char *hist,
                              OneD_dist_profile *distmax, OneD_dist_profile *distmin,
                              int alg_type);

static void hist2D8_fitbound(OneD_dist_profile *distmax,
                             OneD_dist_profile *distmin,
                             unsigned char *hist_area,
                             int alg_type,
                             int sens_adjust,
                             long min_abs_thr);

static int hist2D8_thresh(unsigned char *a, unsigned long tcols_a,
                          unsigned char *b, unsigned long tcols_b,
                          unsigned char *c, unsigned long tcols_c,
                          unsigned long ncols, unsigned long nrows,
```

```
                        unsigned char *hist);

static OneD_dist_profile distmax, distmin;

/*************************************************************************/

int Find2DHistoOutliers(OIP_byte_image *pSrcImage1,
                        OIP_byte_image *pSrcImage2,
                        OIP_byte_image *pDstImageResult,
                        PIXEL_diff_para PixelDiffPara,
                        unsigned char *hist,
                        unsigned char *hist2DBitMap,
                        unsigned long FovID,
                        long shorttest)
{
    int rc;
        unsigned char *aImage1, *aImage2, *aResult; /* ROI pointer */
        const long Passes = HIST_MORPH_PASSES; /*1=3x3 open operation, 2=5x5, etc*/
        char name[128];


        /* check if the ROIs have the same size */
        if ((pSrcImage1->Info.RegWidth != pSrcImage2->Info.RegWidth)
            || (pSrcImage1->Info.RegWidth != pDstImageResult->Info.RegWidth)
            || (pSrcImage1->Info.RegHeight != pSrcImage2->Info.RegHeight)
            || (pSrcImage1->Info.RegHeight != pDstImageResult->Info.RegHeight))
                return -1;

        /* determine ROI pointers */
        aImage1 = pSrcImage1->BufPtr
                + pSrcImage1->Info.RegX0
                + pSrcImage1->Info.RegY0 * pSrcImage1->Info.Width;

        aImage2 = pSrcImage2->BufPtr
                + pSrcImage2->Info.RegX0
                + pSrcImage2->Info.RegY0 * pSrcImage2->Info.Width;

        aResult = pDstImageResult->BufPtr
                + pDstImageResult->Info.RegX0
                + pDstImageResult->Info.RegY0 * pDstImageResult->Info.Width;

        memset(hist, 0, GRAY_LEVELS * GRAY_LEVELS * sizeof(unsigned char));
        hist2D8(aImage1, pSrcImage1->Info.Width,
                aImage2, pSrcImage2->Info.Width,
                pDstImageResult->Info.RegWidth,
                pDstImageResult->Info.RegHeight,
                hist);

        if (shorttest)
        {
            sprintf(name, "../data/2Dhisto_%ld.pgm", FovID);
            printf("Writing 2Dhisto to %s\n",name);fflush(stdout);
            rc = write_image(name, hist, 256, 256);
            if (rc)  ERROR_RETURN(rc, "write 2Dhisto failed");
        }

        /*filter the table (open operation) to delete scattered points*/
        hist2D8_open(hist, hist2DBitMap, Passes);

        /*create two 1D width-profiles for the 2dscatter and their moving averages*/
        hist2D8_1Dprofile(hist, &distmax, &distmin, ALG_FITMASK);

        /*perform thresholding based on the AlgType (1=cloud mask, 2=fit model)*/
        /*create a 2d scatter boundary with the fit model*/
        PREFIX(hist2D8_fitbound)(&distmax, &distmin, hist,
                                 ALG_FITMASK, PixelDiffPara.SensAdjust,
                                 PixelDiffPara.byMinAbsDifference);

        if (shorttest)
        {
            sprintf(name, "../data/2Dhisto_LUT_%ld.pgm", FovID);
            printf("Writing 2Dhisto LUT to %s\n",name);fflush(stdout);
            rc = write_image(name, hist, 256, 256);
            if (rc)  ERROR_RETURN(rc, "write 2Dhisto LUT failed");
        }
```

```
        rc = PREFIX(hist2D8_thresh)(aImage1, pSrcImage1->Info.Width,
                                    aImage2, pSrcImage2->Info.Width,
                                    aResult, pDstImageResult->Info.Width,
                                    pDstImageResult->Info.RegWidth,
                                    pDstImageResult->Info.RegHeight,
                                    hist);
        if (rc) return -1;

        return 0;
}

/*****************************************************************************/
void hist2D8(unsigned char *a, unsigned long tcols_a,
             unsigned char *b, unsigned long tcols_b,
             unsigned long ncols, unsigned long nrows,
             unsigned char *table)

/***********************************************
        a,b -- TCL of ROI (images)
        tcols_a -- image width
        ncols, nrows -- ROI size
        table -- scatter table (a 256x256 image)
 ***********************************************/
{
        unsigned long i, j;
        unsigned long skip_a, skip_b;

        unsigned long index;

        skip_a = tcols_a - ncols;
        skip_b = tcols_b - ncols;

        for (i = 0; i < nrows; ++i)
        {
                for (j = 0; j < ncols; ++j)
                {
                        index=((*a++)<<GRAY_SCALE_BITS)+(*b++);
                        table[index] = 1;
                }

                a += skip_a;
                b += skip_b;
        }

}


void PREFIX(hist2D8_open)(unsigned char *hist, unsigned char *histBitMap, long Passes)
{
        OIP_bit_image  pSrcBitImage; /*, pDstBitImage;*/
        MORPH_setup           pMorphSetup;


        pSrcBitImage.BufPtr = histBitMap;
        pSrcBitImage.Info.Width              = GRAY_LEVELS;
        pSrcBitImage.Info.Height       = GRAY_LEVELS;
        pSrcBitImage.Info.RegX0      = 0;
        pSrcBitImage.Info.RegY0      = 0;
        pSrcBitImage.Info.RegWidth   = GRAY_LEVELS;
        pSrcBitImage.Info.RegHeight    = GRAY_LEVELS;

        binarize8(hist,GRAY_LEVELS, pSrcBitImage.BufPtr, GRAY_LEVELS, GRAY_LEVELS);


        morph_setup1(Passes, GRAY_LEVELS, GRAY_LEVELS, &pMorphSetup);
        if (pMorphSetup == NULL)
                fprintf(stderr,"morph_setup1\n");

        close1(pSrcBitImage.BufPtr, pSrcBitImage.BufPtr, /*pDstBitImage.BufPtr,*/
               pSrcBitImage.Info.RegWidth,
               pSrcBitImage.Info.RegHeight,
               Passes,
               pMorphSetup);

        open1(pSrcBitImage.BufPtr, pSrcBitImage.BufPtr, /*pDstBitImage.BufPtr,*/
```

```
            pSrcBitImage.Info.RegWidth,
            pSrcBitImage.Info.RegHeight,
            Passes,
            pMorphSetup);

        unpack(pSrcBitImage.BufPtr, GRAY_LEVELS, GRAY_LEVELS, hist, GRAY_LEVELS);

        morph_cleanup1(&pMorphSetup);

}

void hist2D8_1Dprofile(unsigned char *hist,
                       OneD_dist_profile *distmax,
                       OneD_dist_profile *distmin,
                       int alg_type)
{
        int i, j, index, temp;
        float temp1=0, temp2=0;
        long dist_init[GRAY_LEVELS];

        for (i = 0; i < GRAY_LEVELS; ++i)
        {
                distmax->buf[i] = 0;
                distmin->buf[i] = 0;
                distmax->mva[i] = 0;
                distmin->mva[i] = 0;
                distmax->model[i] = 0;
                distmin->model[i] = 0;
                dist_init[i] = 0;
        }
        for (i = 0; i < GRAY_LEVELS; ++i)
        {
                for (j = 0; j < GRAY_LEVELS; ++j)
                {
                        index = (i << GRAY_SCALE_BITS) + j;
                        if (hist[index])
                        {
                                index = (long)((i + j) / 2. + 0.5);
                                temp = i - j;
                                /*if first item for this index, assign both max and min*/
                                if (!dist_init[index])
                                {
                                        dist_init[index]++;
                                        distmax->buf[index] = (int)(temp + 0.5);
                                        distmin->buf[index] = (int)(temp + 0.5);
                                }
                                else
                                {
                                /*temp = distance( table_bin[index], 45degreeline);*/
                                if (temp > distmax->buf[index]) /* (ref - test) envelope*/
                                        distmax->buf[index] = (int)(temp + 0.5);
                                if (temp < distmin->buf[index]) /* (test - ref) envelope*/
                                        distmin->buf[index] = (int)(temp + 0.5);
                                }
                        }
                }
        }
        /*need to divide distminbuf & distmaxbuf by sqrt2*/
        for (i = 0; i < GRAY_LEVELS; ++i)
        {
                distmax->buf[i] = (int) (distmax->buf[i] / SQRT2 + 0.5);
                distmin->buf[i] = (int) (distmin->buf[i] / SQRT2 + 0.5);
        }

        /*Remove invalid data points from the beginning and end of
          the profile: All points below the offset and all points above where
          the profile starts to shrink are set to invalid*/
        /*First set the validity attribute to 1 for all points on the profile*/
        for (i = 0; i < GRAY_LEVELS; ++i)
        {
                distmax->valid[i] = 1;
                distmin->valid[i] = 1;
        }
        /*Find the offset, i.e. the first point w/ distance > MIN_DIST*/
        i = 0;
        while (abs(distmax->buf[i]) < MIN_DIST && i < GRAY_LEVELS)
```

```
                distmax->valid[i++] = 0;

        distmax->offset = i;
        i = 0;
        while (abs(distmin->buf[i]) < MIN_DIST && i < GRAY_LEVELS)
                distmin->valid[i++] = 0;

        distmin->offset = i;
        /*fprintf(stderr,"distmax.offset = %d, distmin.offset = %d\n",
                                                distmax->offset, distmin->offset);*/
        if (alg_type == ALG_FITMASK)
        {
                i = GRAY_LEVELS -1;
                while (abs(distmax->buf[i]) < MIN_DIST && i > distmax->offset)
                        distmax->valid[i--] = 0;

                distmax->endpoint = i;
                i = GRAY_LEVELS -1;
                while (abs(distmin->buf[i]) < MIN_DIST && i > distmin->offset)
                        distmin->valid[i--] = 0;

                distmin->endpoint = i;
        }
        /*Calculate a moving avg to smooth each profile*/
        index = (int)(MVA_SIZE / 2.0);
        for (i = 0; i < index; ++i)
        {
                temp1 += distmax->buf[i];
                temp2 += distmin->buf[i];
        }
        temp1 = temp1 / MVA_SIZE;
        temp2 = temp2 / MVA_SIZE;
        for (i = index; i < MVA_SIZE; ++i)
        {
                temp1 += distmax->buf[i];
                distmax->mva[i-index] = (int)(temp1 / MVA_SIZE + 0.5);
                temp2 += distmin->buf[i];
                distmin->mva[i-index] = (int)(temp2 / MVA_SIZE + 0.5);
        }
        temp1 = temp1 / MVA_SIZE;
        temp2 = temp2 / MVA_SIZE;
        for (i = MVA_SIZE; i < GRAY_LEVELS; ++i)
        {
                temp1 += (float)(distmax->buf[i] - distmax->buf[i-MVA_SIZE]) / MVA_SIZE;
                distmax->mva[i-index] = (int)(temp1 + 0.5);
                temp2 += (float)(distmin->buf[i] - distmin->buf[i-MVA_SIZE]) / MVA_SIZE;
                distmin->mva[i-index] = (int)(temp2 + 0.5);
        }
        for (i = GRAY_LEVELS; i < (GRAY_LEVELS+index); ++i)
        {
                temp1 += (float)( - distmax->buf[i-MVA_SIZE]) / MVA_SIZE;
                distmax->mva[i-index] = (int)(temp1 + 0.5);
                temp2 += (float)( - distmin->buf[i-MVA_SIZE]) / MVA_SIZE;
                distmin->mva[i-index] = (int)(temp2 + 0.5);
        }
/*replace the mva values at the first and last MVA_SIZE/2 points w/ orig buf values*/
        for (i = distmax->offset; i < distmax->offset + index + 1; i++)
                distmax->mva[i] = distmax->buf[i];
        for (i = distmax->endpoint - index; i < distmax->endpoint + 1; i++)
                distmax->mva[i] = distmax->buf[i];
        for (i = distmin->offset; i < distmin->offset + index + 1; i++)
                distmin->mva[i] = distmin->buf[i];
        for (i = distmin->endpoint - index; i < distmin->endpoint + 1; i++)
                distmin->mva[i] = distmin->buf[i];

        distmax->count = distmax->endpoint - distmax->offset;
        distmin->count = distmin->endpoint - distmin->offset;

        for (i = distmax->endpoint; i < GRAY_LEVELS; ++i)
                distmax->valid[i] = 0;
        for (i = distmin->endpoint; i < GRAY_LEVELS; ++i)
                distmin->valid[i] = 0;
}
void PREFIX(hist2D8_fitbound)(OneD_dist_profile *distmax, OneD_dist_profile *distmin,
unsigned char *hist_area, int alg_type, int sens_adjust, long min_abs_thr)
{
```

```
        int i,j, index, axis1;
        double temp;
        int openmargin, minopenmargin;
        long offset, endpoint;

        openmargin = 2 * HIST_MORPH_PASSES + 1 - sens_adjust;
        minopenmargin = min_abs_thr; /* + openmargin;*/
        offset = (distmin->offset < distmax->offset) ? distmin->offset : distmax->offset;
        endpoint = (distmin->endpoint < distmax->endpoint) ? distmin->endpoint :
                   distmax->endpoint;
        if (offset < 1) offset = 1;
        if (endpoint > GRAY_LEVELS - 2) endpoint = GRAY_LEVELS - 2;
switch (alg_type)
        {
        case ALG_FITMASK:
          /*    for (i = distmax->offset - 1; i < distmax->endpoint + 1; i++)
                      distmax->model[i] = distmax->mva[i] + openmargin;
                for (i = distmin->offset - 1; i < distmin->endpoint + 1; i++)
                      distmin->model[i] = distmin->mva[i] - openmargin;
          */
                for (i = offset - 1; i < endpoint + 2; ++i)
                {
                  distmax->model[i] = distmax->mva[i] + openmargin;
                  distmin->model[i] = distmin->mva[i] - openmargin;
                }
                for (i = 0; i < GRAY_LEVELS; ++i)
                {
                        for (j = 0; j < GRAY_LEVELS; ++j)
                        {
                                index = (i<<GRAY_SCALE_BITS)+j;
                                hist_area[index] = 1;
                                axis1 = (int)(((i + j) >> 1) + 0.5);
                                temp = (i - j) / SQRT2;
                                if (abs(temp) < minopenmargin)
                                  hist_area[index] = 0;
                                if ((temp < distmax->model[axis1]) &&
                                    (temp > distmin->model[axis1]))
                                                hist_area[index] = 0;
                        }
                }
                break;
        default:
                break;
        }
}

static int PREFIX(hist2D8_thresh)(unsigned char *a, unsigned long tcols_a,
                        unsigned char *b, unsigned long tcols_b,
                        unsigned char *c, unsigned long tcols_c,
                        unsigned long ncols, unsigned long nrows,
                        unsigned char *hist)
{
        unsigned long i, j, index;
        unsigned long skip_a, skip_b, skip_c;

        skip_a = tcols_a - ncols;
        skip_b = tcols_b - ncols;
        skip_c = tcols_c - ncols;

        for (i = 0; i < nrows; ++i)
        {
          for (j = 0; j < ncols; ++j)
          {
            index=((*a++)<<GRAY_SCALE_BITS)+(*b++);
            *c++ = hist[index];
          }
          a += skip_a;
          b += skip_b;
          c += skip_c;
        }
        return 0;
}
```

REVENDICATIONS

1. Procédé de détection de défauts, caractérisé en ce qu'il comprend les étapes consistant à :

(a) fournir (310) une première image d'un objet à inspecter et une deuxième image associée ;

(b) aligner (320) la première image avec la deuxième image ;

(c) créer (330) un premier tracé en traçant les niveaux de gris des pixels provenant de la première image en fonction des niveaux de gris des pixels correspondants provenant de la deuxième image ;

(d) créer (340) un deuxième tracé en filtrant le premier tracé ;

(e) créer (350) un masque tel que le profil du masque est défini par la configuration du deuxième tracé ; et

(f) utiliser (380) le masque pour détecter des défauts représentés dans la première image.

2. Procédé selon la revendication 1, caractérisé en ce qu'un filtrage (340) est exécuté en utilisant un filtre morphologique.

3. Procédé selon la revendication 1, caractérisé en ce qu'une extension du masque peut être ajustée (360) par l'utilisateur.

4. Procédé selon la revendication 1, caractérisé en ce qu'il comprend en outre l'utilisation d'un filtre à moyenne mobile pour lisser le profil du masque.

5. Procédé selon la revendication 1, caractérisé en ce que la deuxième image est obtenue à partir d'une base de données.

6. Procédé selon la revendication 1, caractérisé en ce qu'il comprend en outre une mémorisation des tracés résultant des étapes (c) et (d) dans un support lisible par informatique.

7. Support lisible par informatique caractérisé en ce qu'il mémorise un programme de mise en œuvre du procédé de la revendication 1.

8. Support lisible par informatique, caractérisé en ce qu'il comprend :

— une série d'emplacements de mémoire qui contiennent des données représentant une première image et une deuxième image qui lui est associée, lesdites première et deuxième image incluant chacune une série de pixels qui sont définis chacun par une coordonnée d'emplacement et par un niveau de gris ; et

— un ensemble qui comprend une série d'emplacements de mémoire mémorisant des données définissant un masque, le masque étant créé en filtrant un tracé des niveaux de gris des pixels provenant de la première image en fonction des niveaux de gris de pixels correspondants provenant de la deuxième image.

9. Support selon la revendication 8, caractérisé en ce que le filtrage est exécuté en utilisant un filtre morphologique.

10. Support selon la revendication 8, caractérisé en ce qu'un algorithme à moyenne mobile est utilisé pour lisser le tracé des niveaux de gris de pixels provenant de la première image en fonction des niveaux de gris de pixels provenant de la deuxième image.

2799027

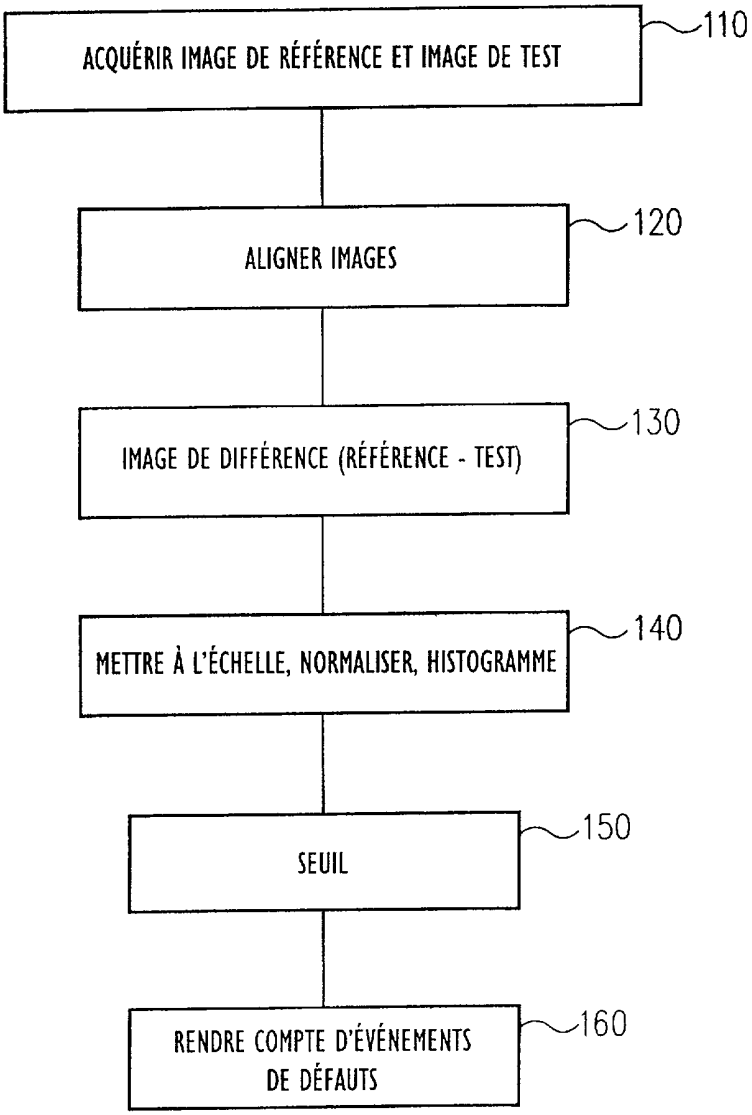


FIG. 1

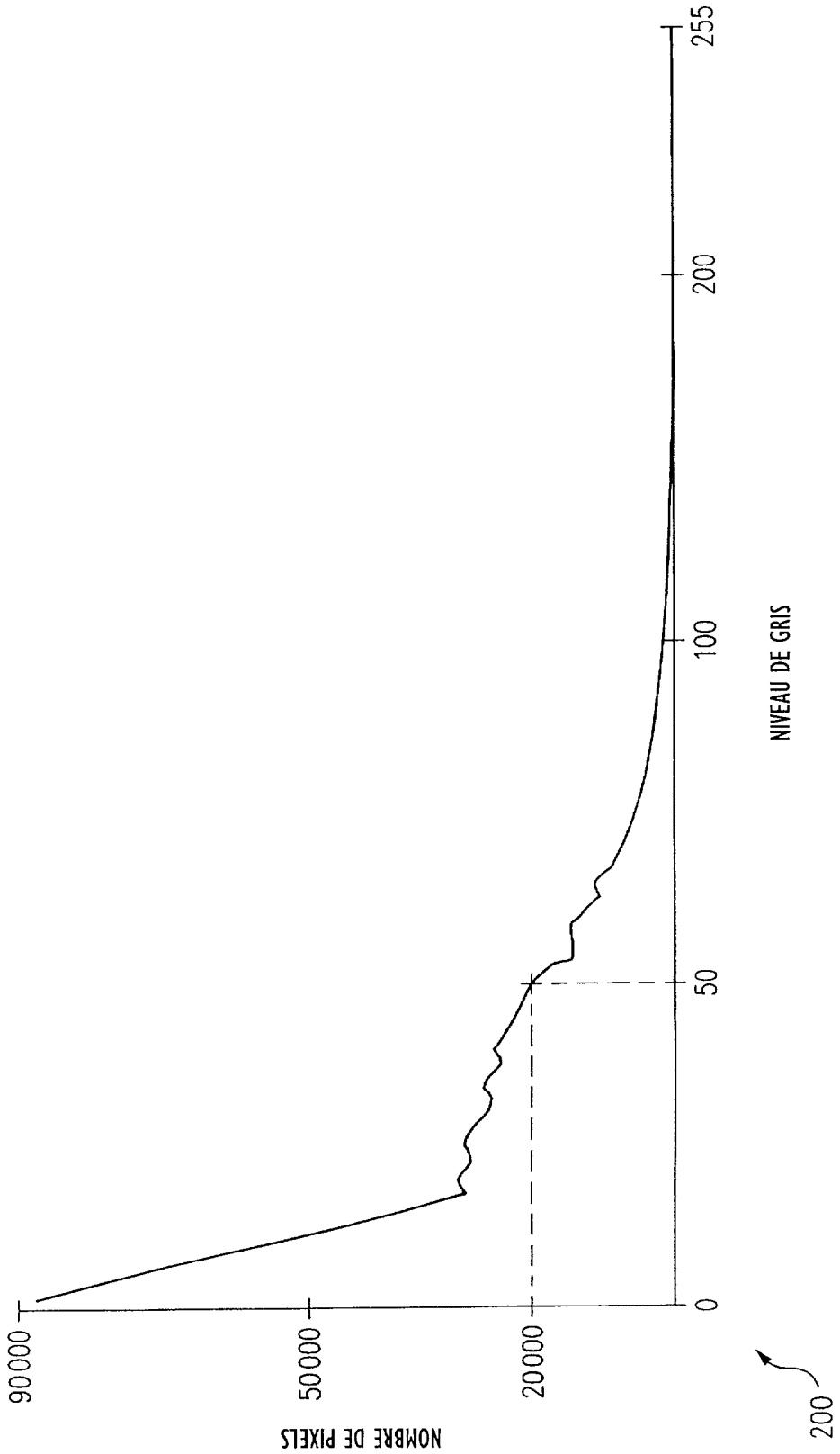

NOMBRE DE PIXELS
90000
50000
20000
0
50
100
200
255
NIVEAU DE GRIS
200

FIG. 2

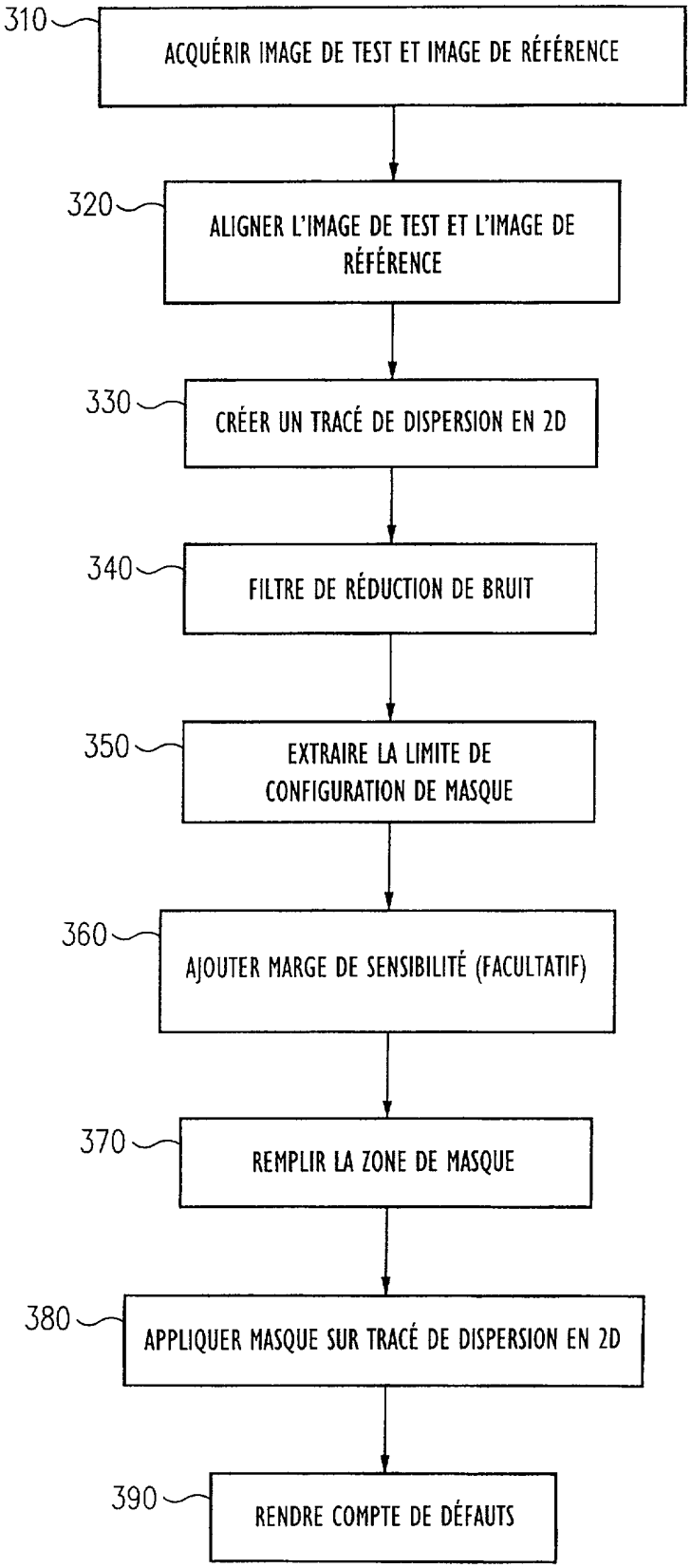

310
ACQUÉRIR IMAGE DE TEST ET IMAGE DE RÉFÉRENCE
320
ALIGNER L'IMAGE DE TEST ET L'IMAGE DE RÉFÉRENCE
330
CRÉER UN TRACÉ DE DISPERSION EN 2D
340
FILTRE DE RÉDUCTION DE BRUIT
350
EXTRAIRE LA LIMITE DE CONFIGURATION DE MASQUE
360
AJOUTER MARGE DE SENSIBILITÉ (FACULTATIF)
370
REMPLIR LA ZONE DE MASQUE
380
APPLIQUER MASQUE SUR TRACÉ DE DISPERSION EN 2D
390
RENDRE COMPTE DE DÉFAUTS

FIG. 3

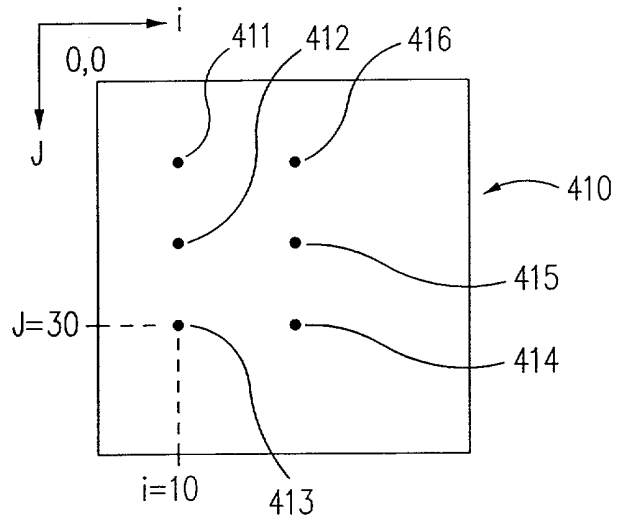


FIG. 4A

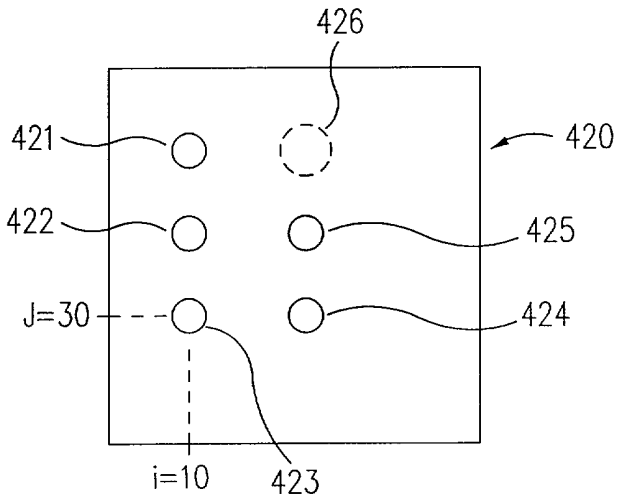


FIG. 4B

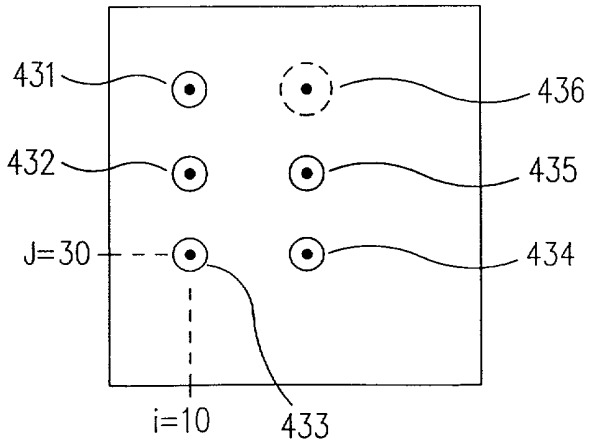


FIG. 4C

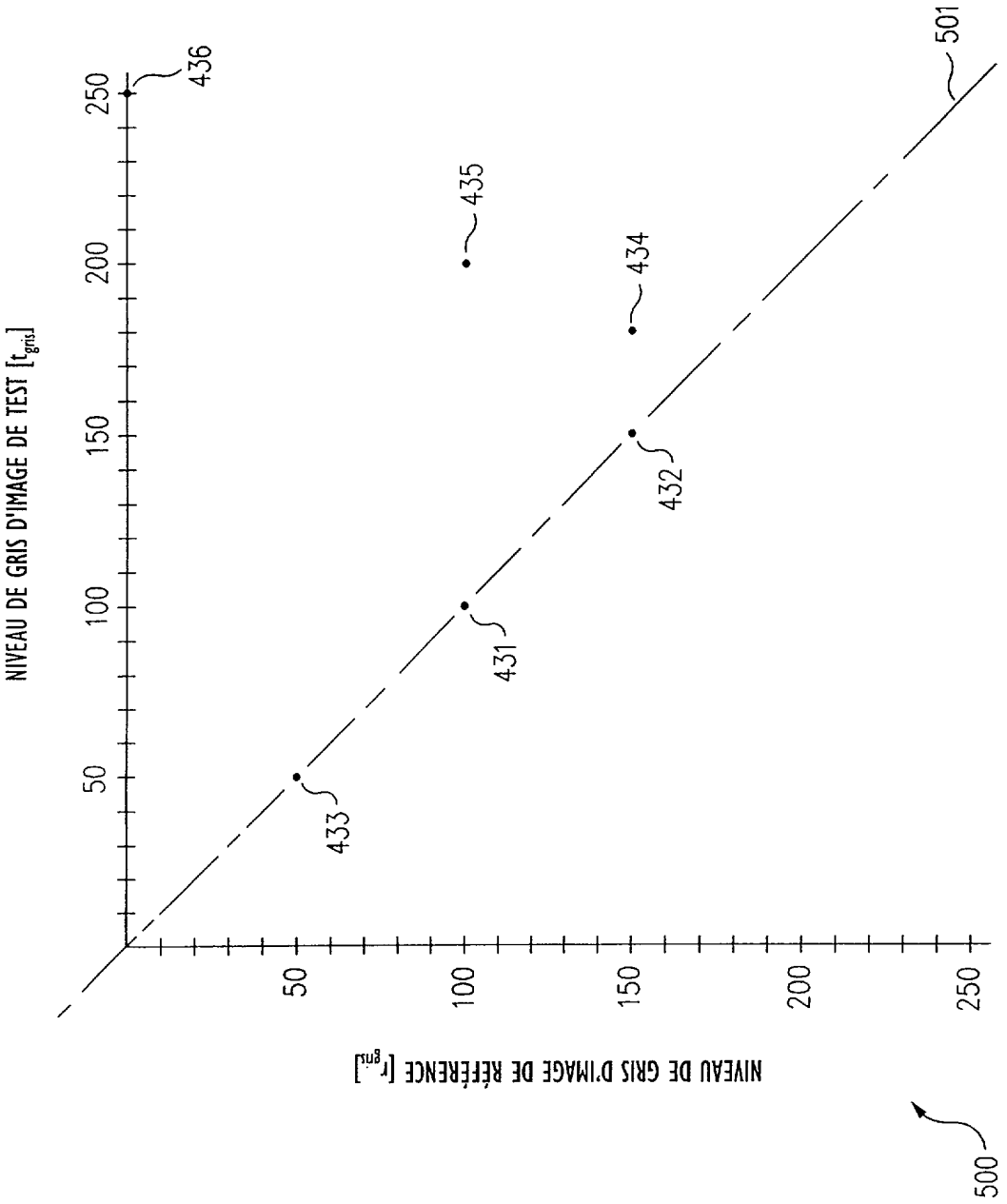

NIVEAU DE GRIS D'IMAGE DE TEST [t_{gris}]
NIVEAU DE GRIS D'IMAGE DE RÉFÉRENCE [r_{gris}]
50
100
150
200
250
433
431
435
434
432
436
501
500

FIG. 5

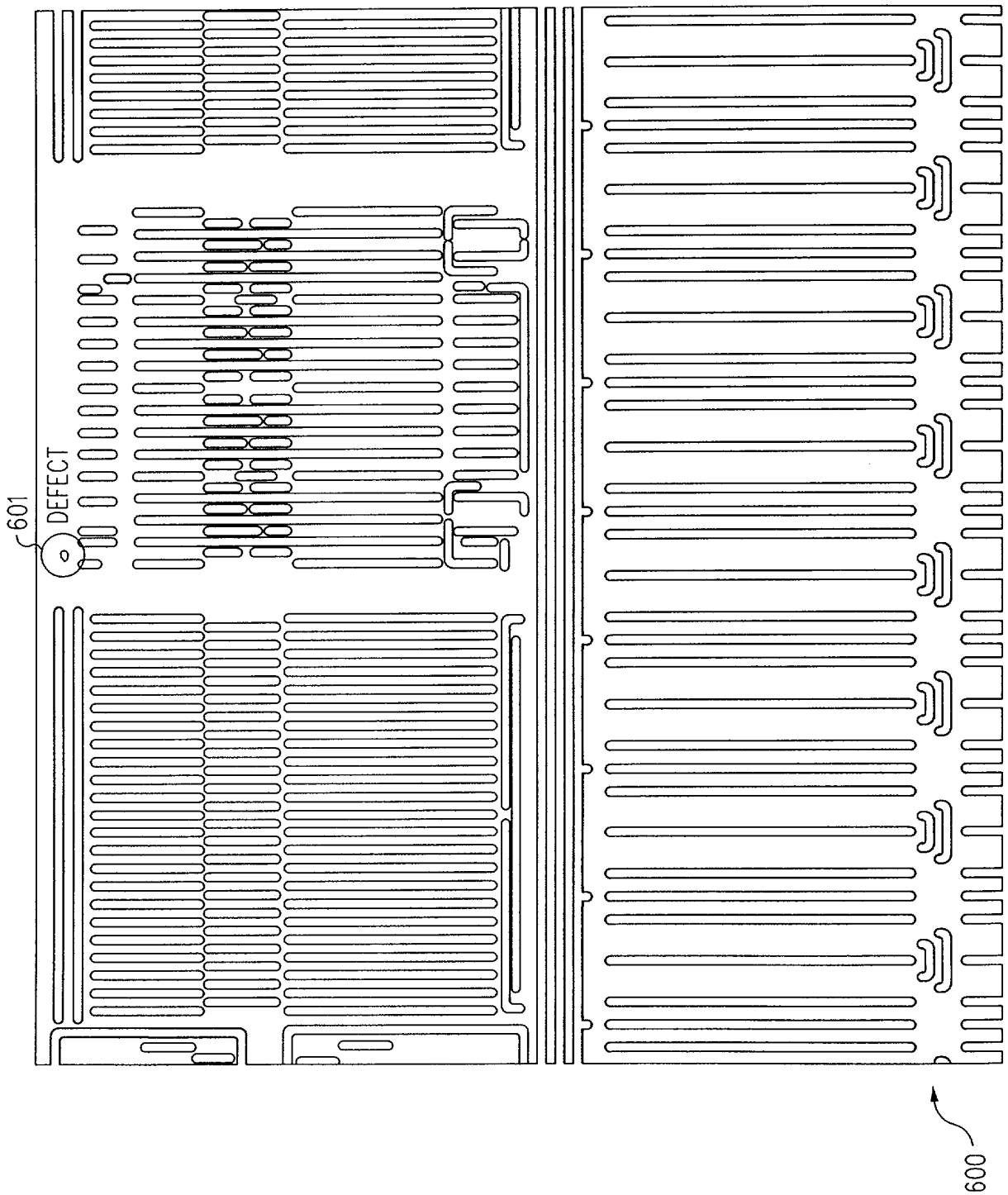


FIG. 6

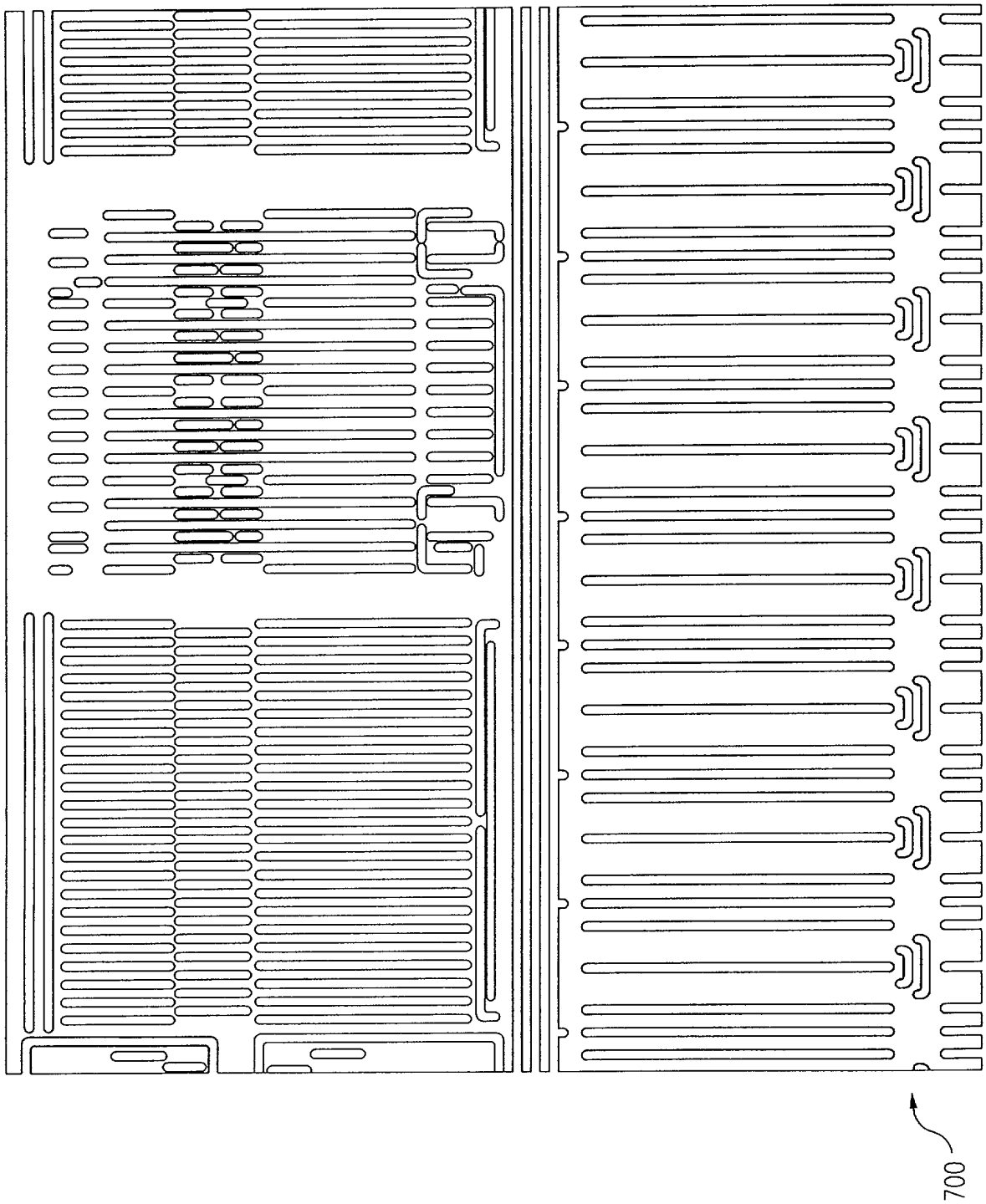


FIG. 7

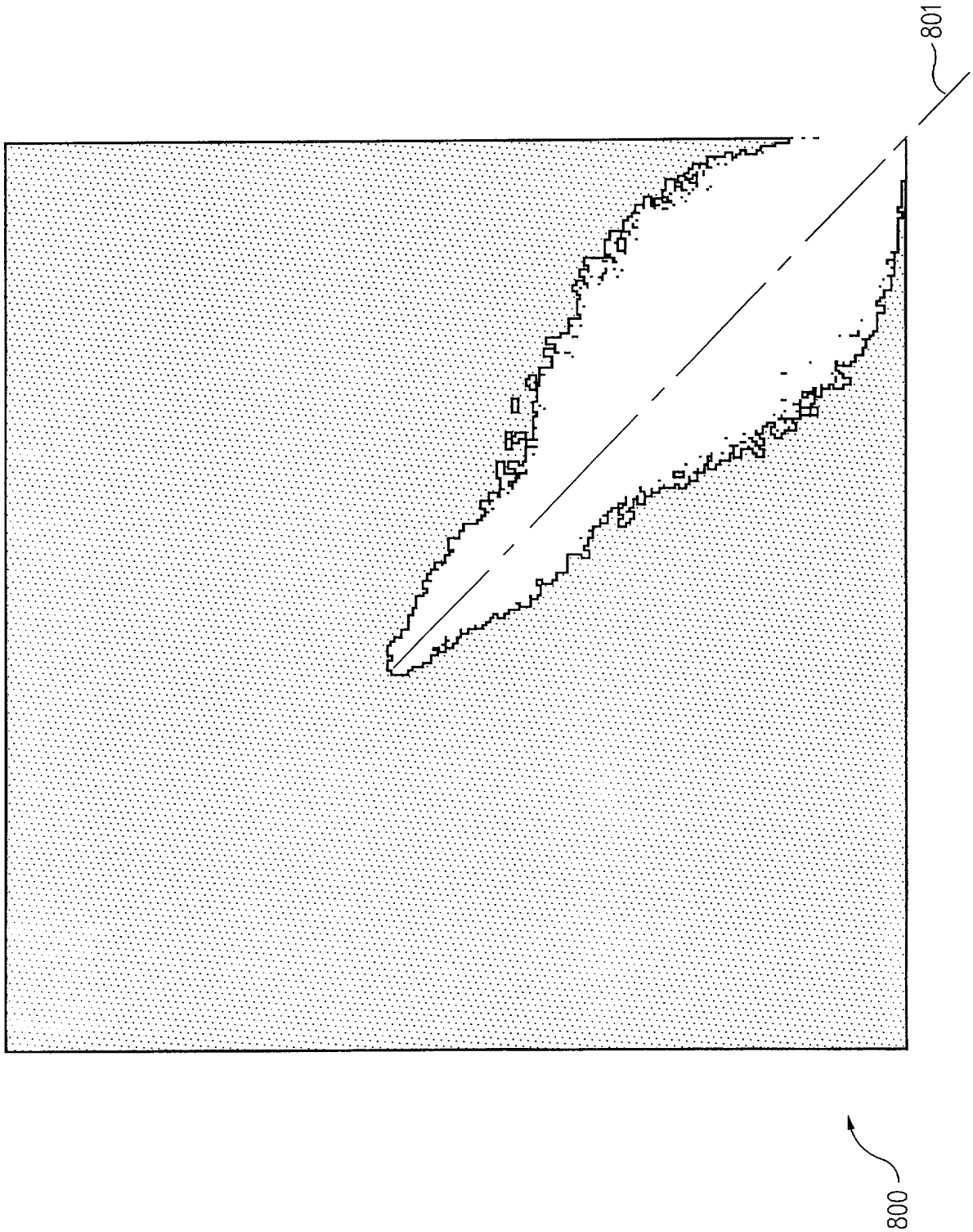


FIG. 8

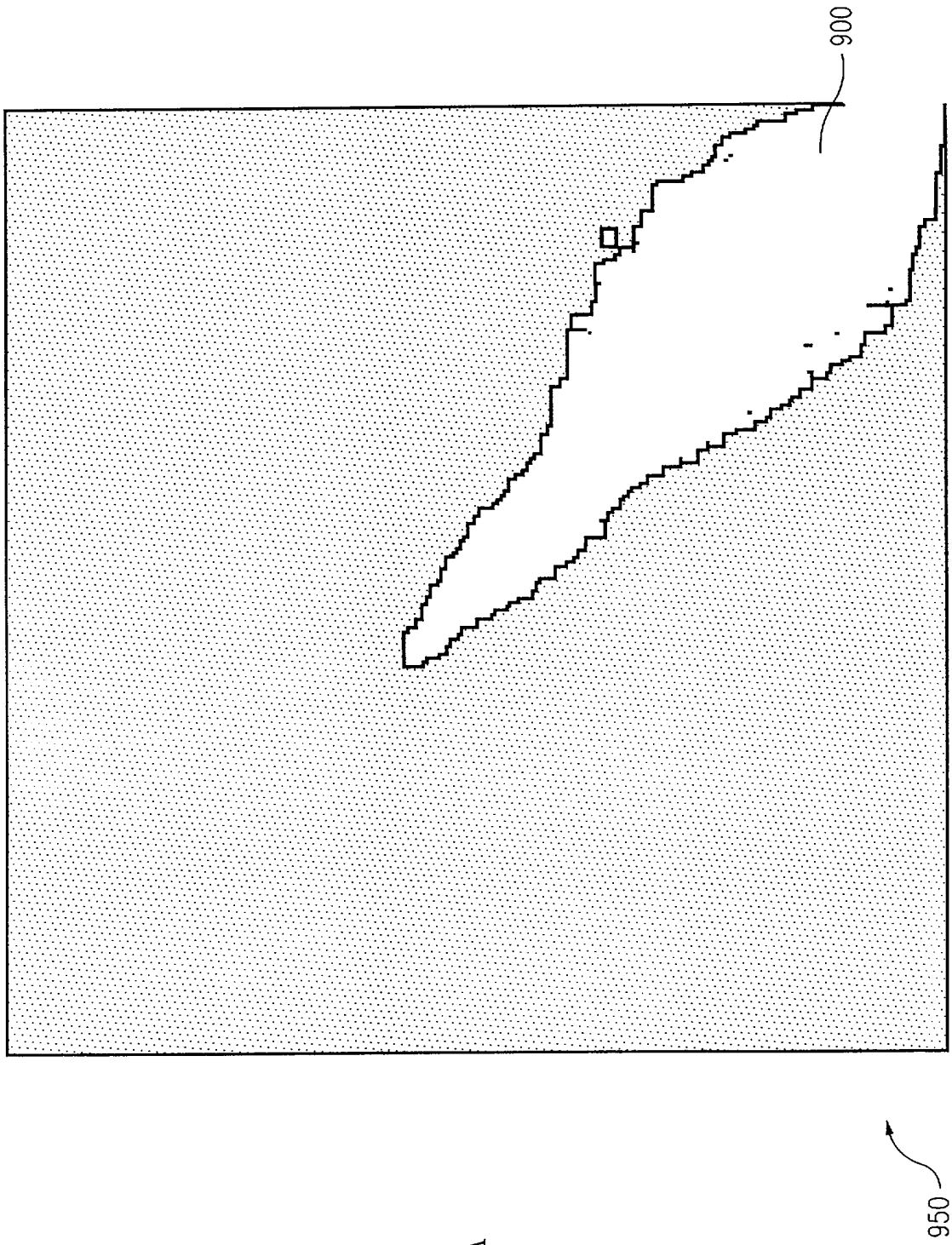


FIG. 9A

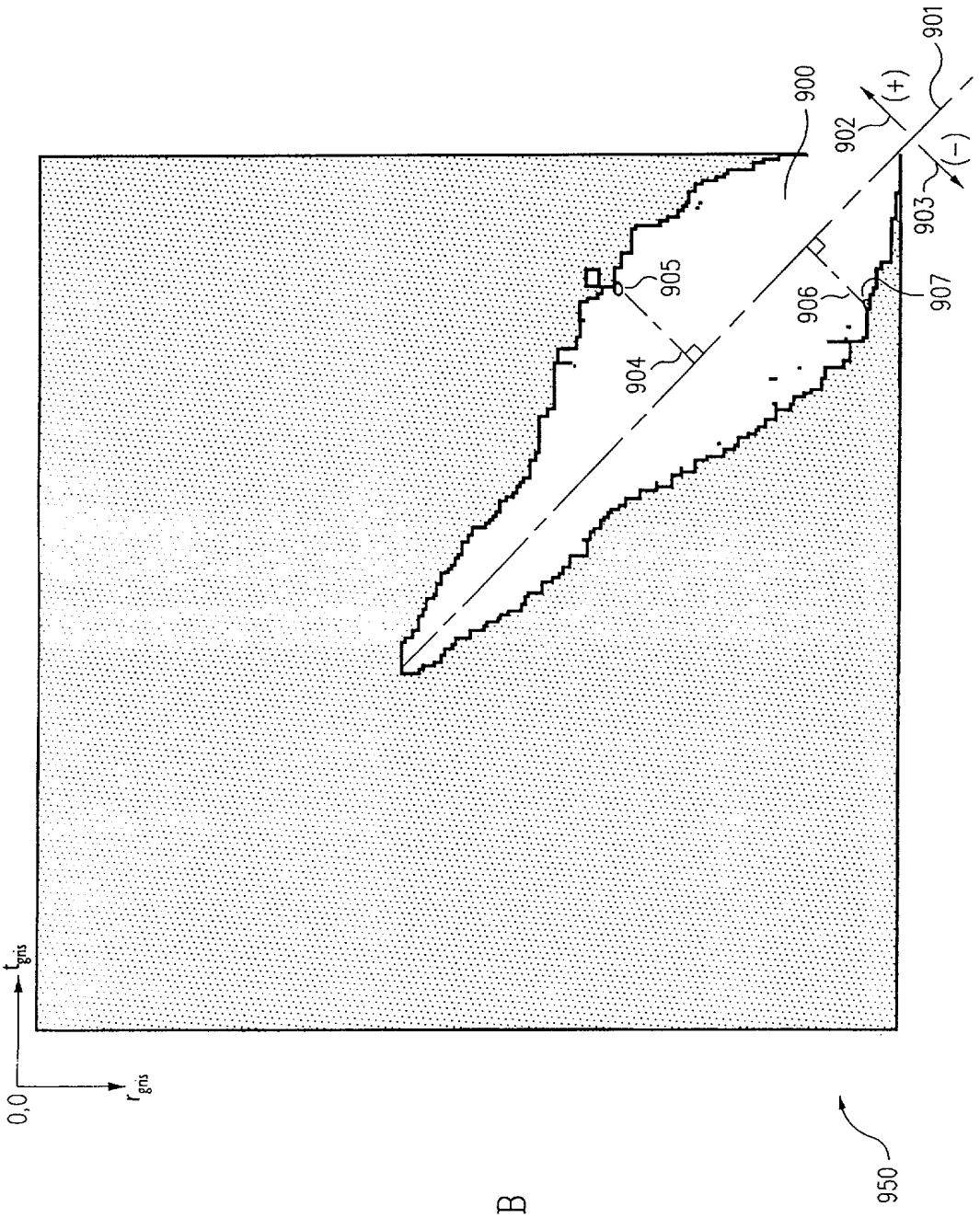


FIG. 9B

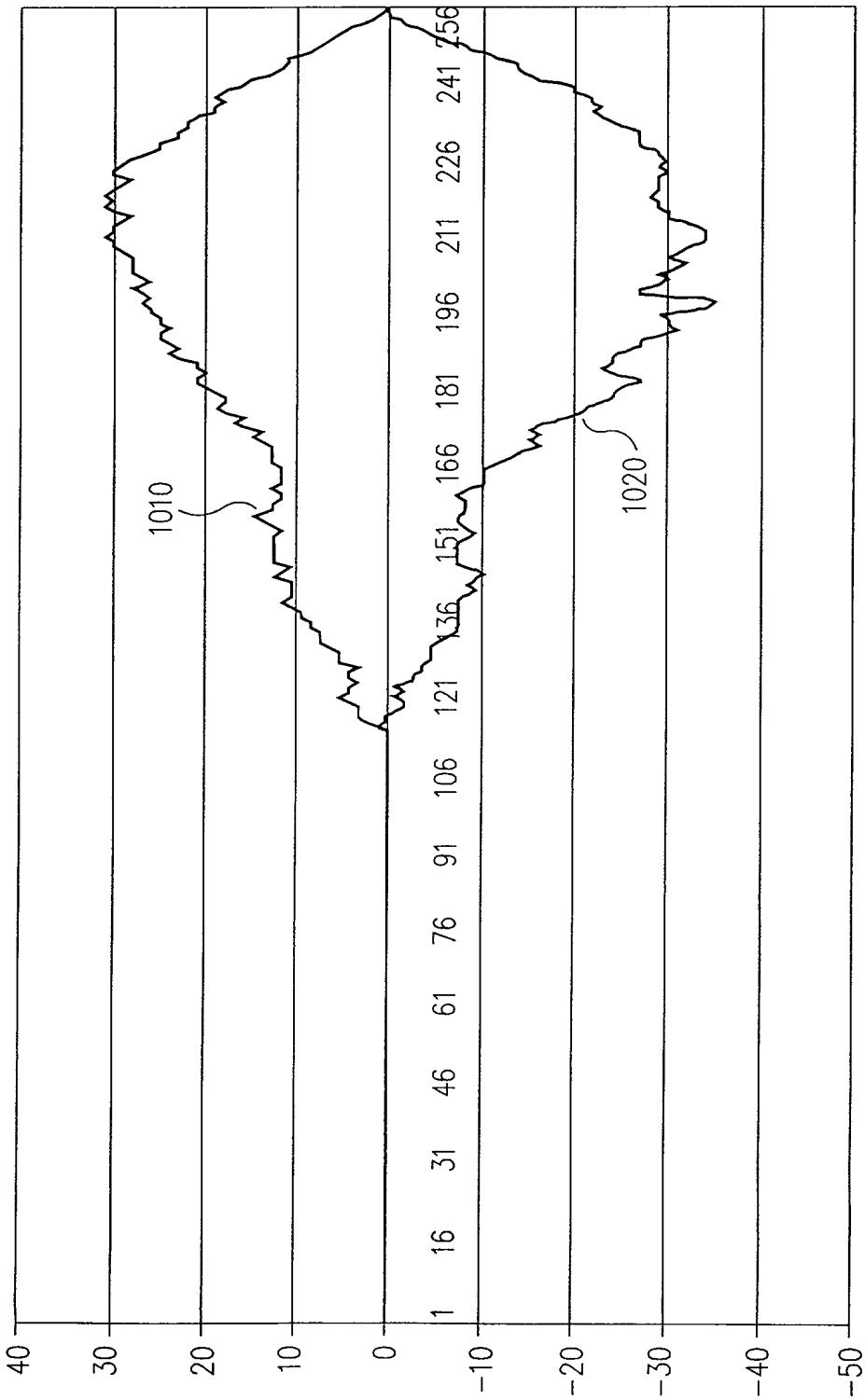

40
30
20
10
0
-10
-20
-30
-40
-50
1
16
31
46
61
76
91
106
121
136
151
166
181
196
211
226
241
256
1010
1020
Rprofil
1000

FIG. 10A

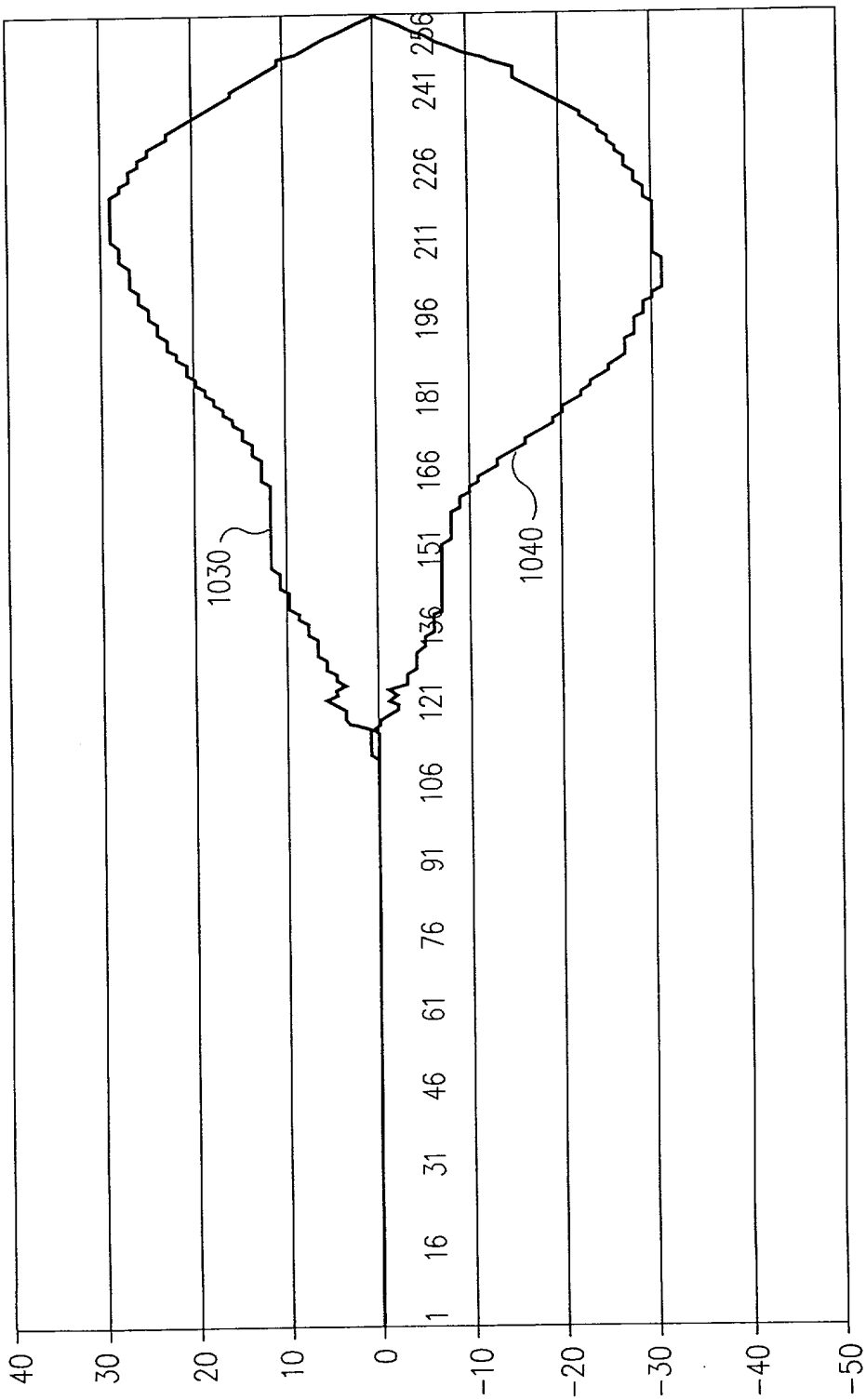

40
30
20
10
0
-10
-20
-30
-40
-50
1
16
31
46
61
76
91
106
121
136
151
166
181
196
211
226
241
256
1030
1040
Rprofil
2000

FIG. 10B

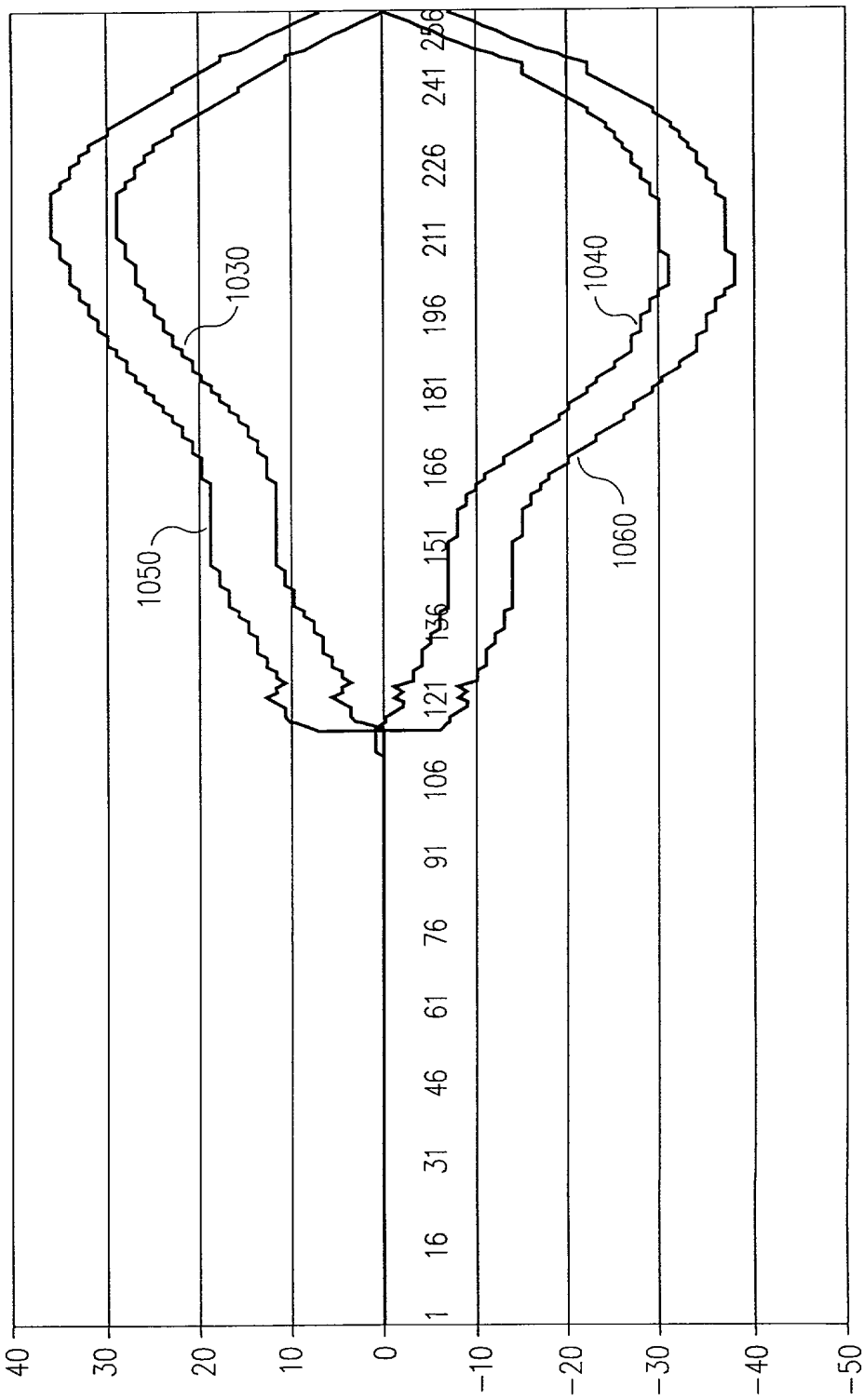

40
30
20
10
0
-10
-20
-30
-40
-50
1
16
31
46
61
76
91
106
121
136
151
166
181
196
211
226
241
256
1030
1040
1050
1060
Rprofil

FIG. 10C

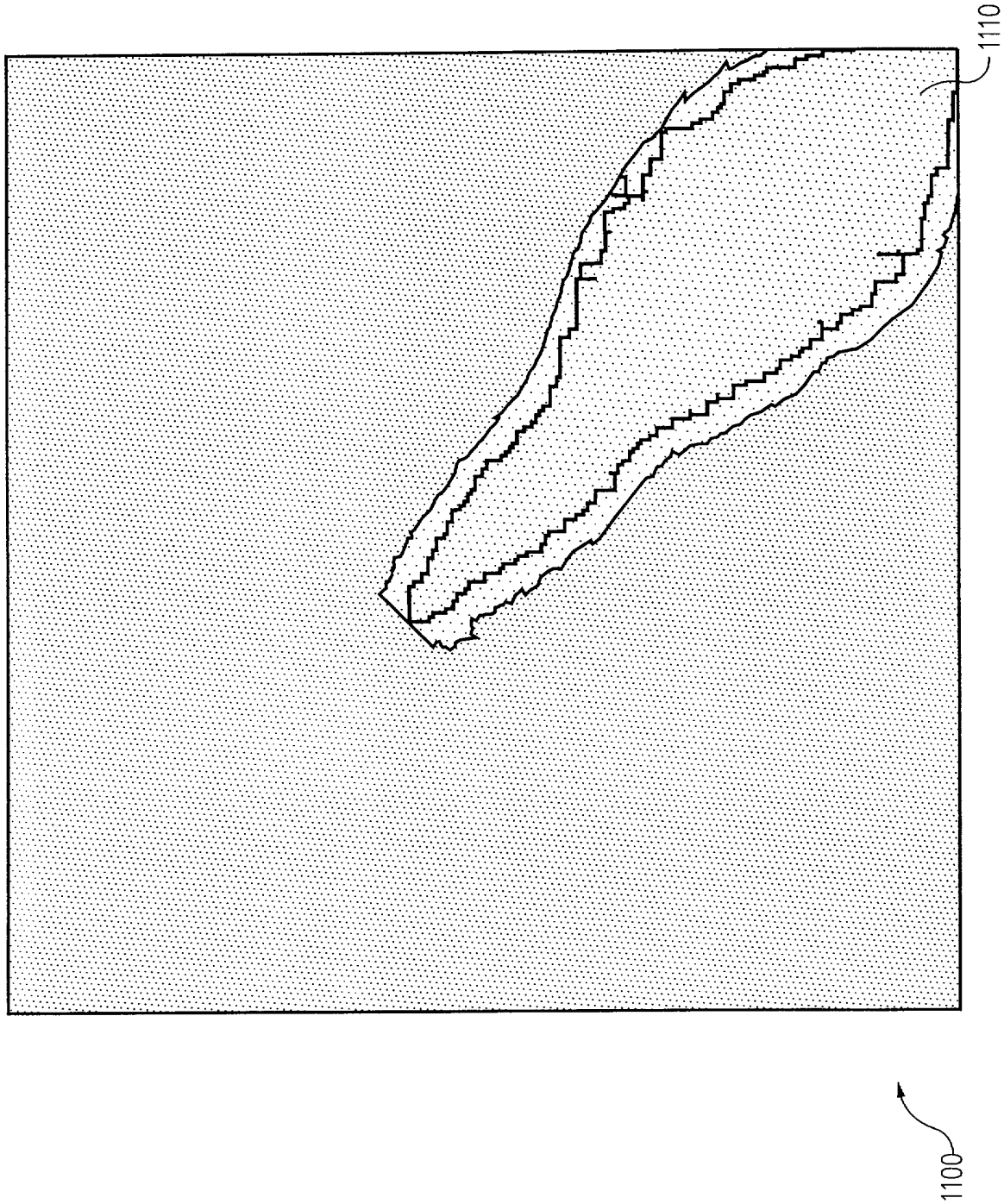


FIG. 11

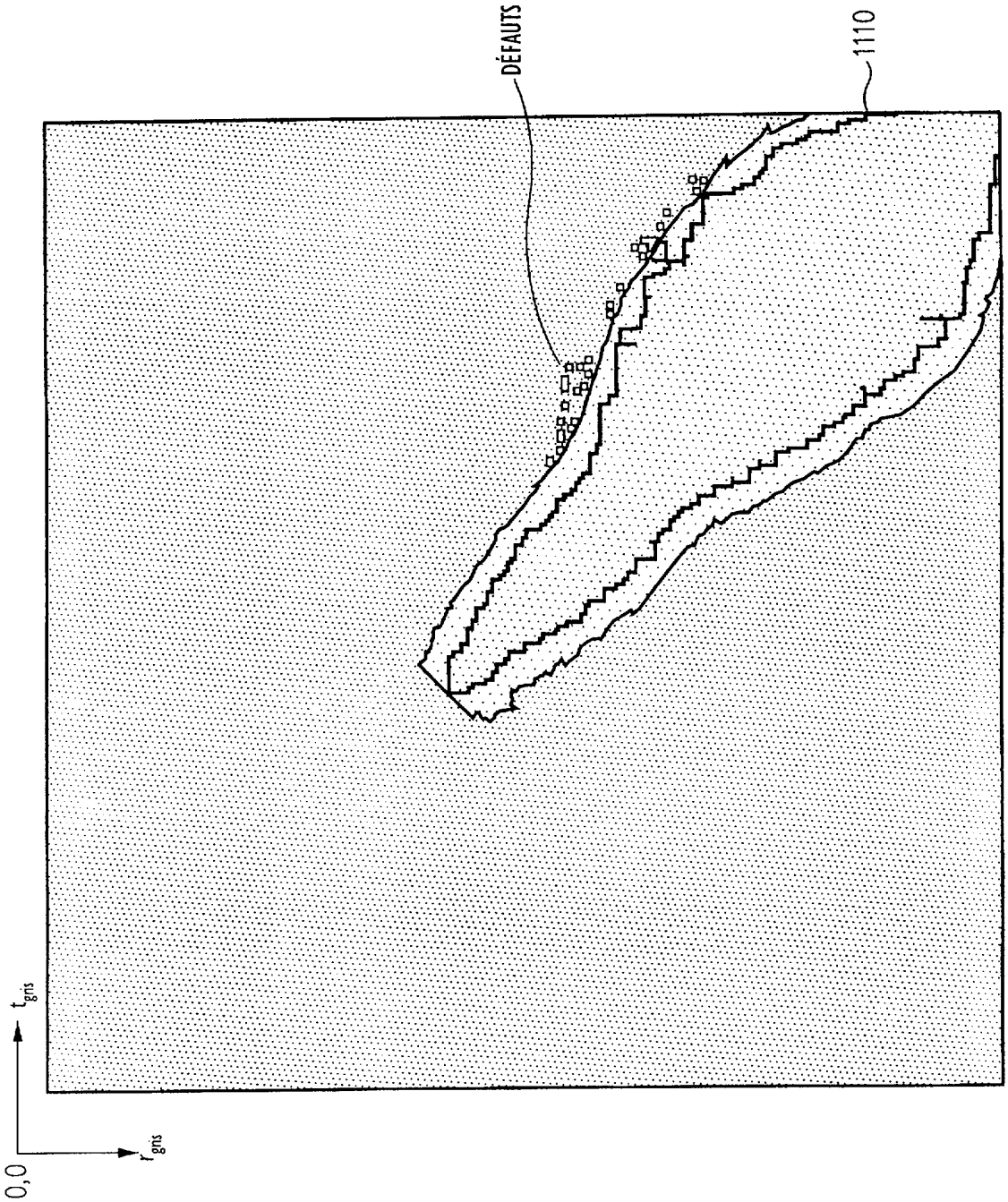


FIG. 12

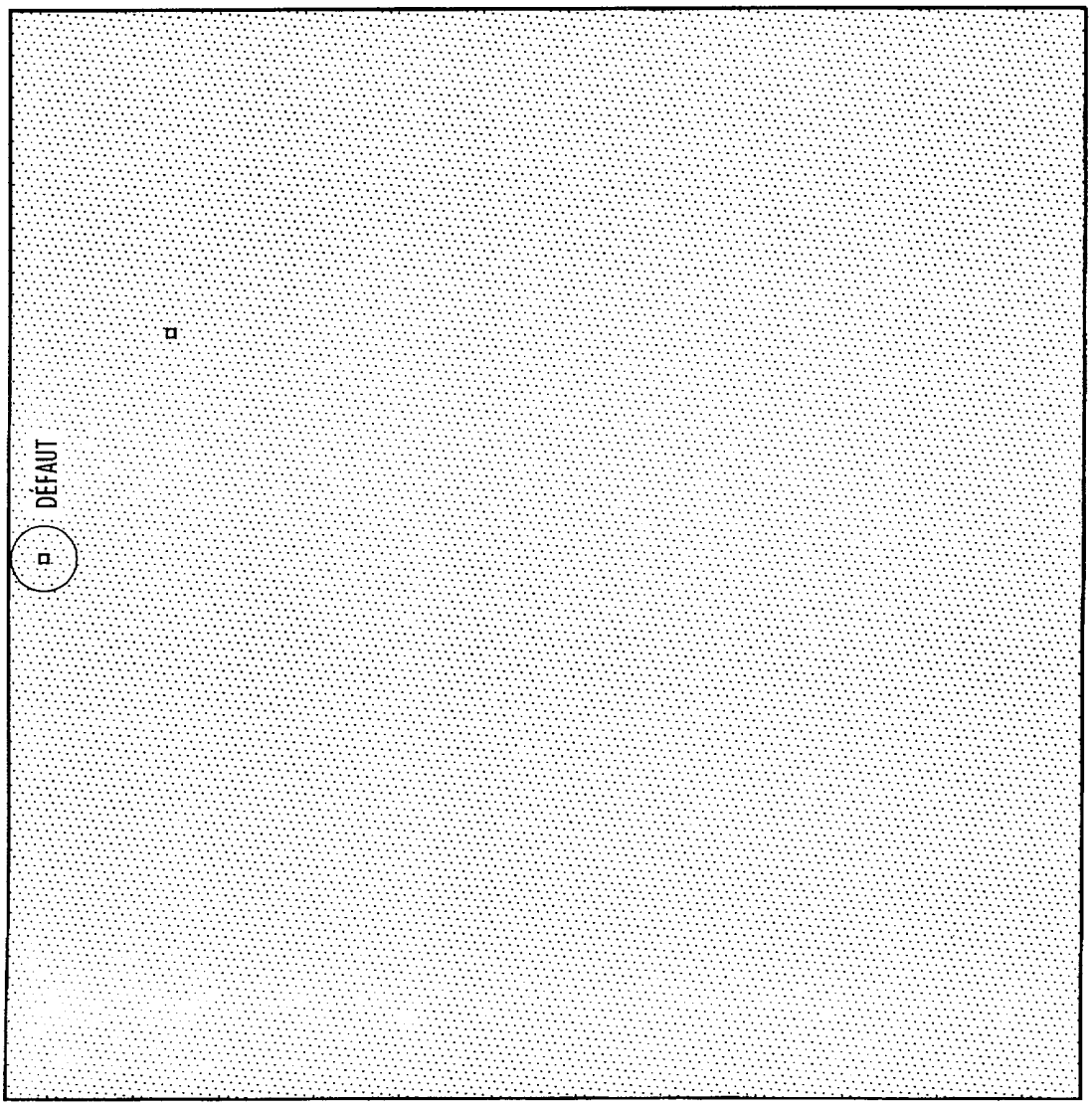


FIG. 13

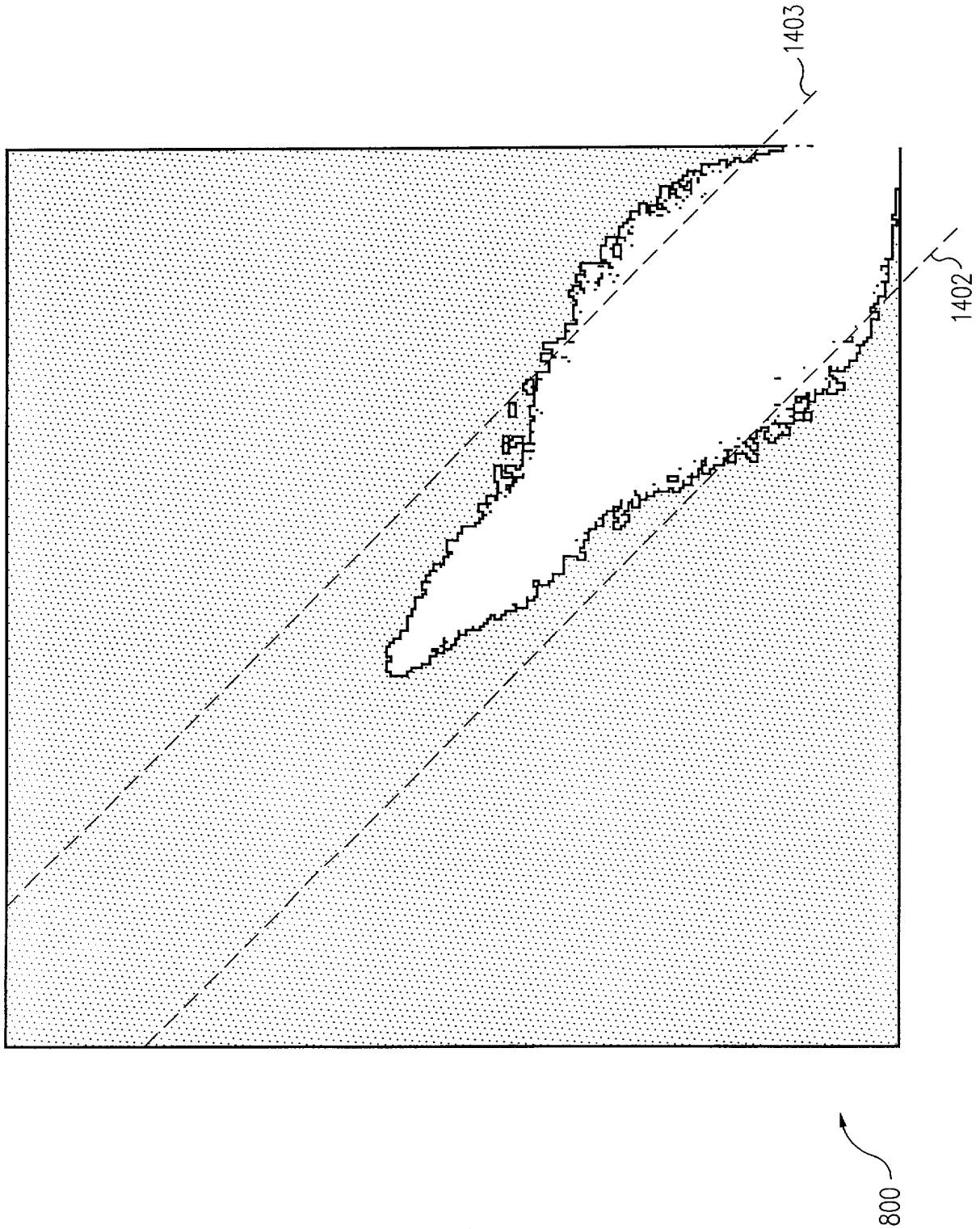


FIG. 14

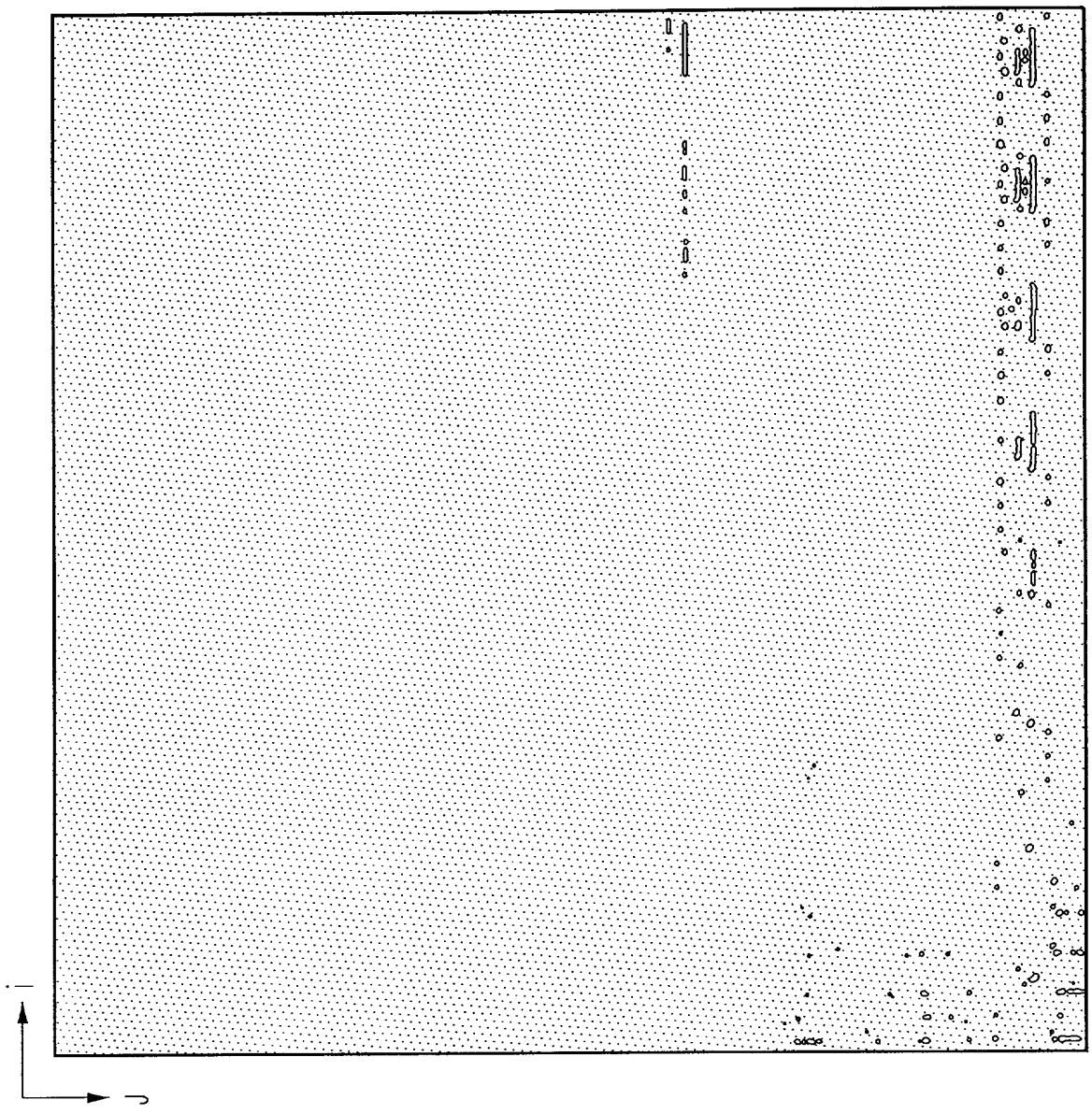


FIG. 15