Extension des segments flous aux images en niveaux de gris pour l'extraction interactive de segments de droites

Philippe Even, Bertrand Kerautret

Extension des segments flous aux images en niveaux de gris pour l'extraction interactive de segments de droites

P. Even B. Kerautret
LORIA
Nancy Université - IUT de Saint Dié des Vosges
{kerautre,even}@loria.fr

Résumé

La reconnaissance de segments de droites discrètes est un problème significatif dans le domaine de la géométrie discrète et dans beaucoup d'applications d'extraction de paramètres géométriques. L'utilisation du concept des segments flous permet de traiter des images binaires bruitées [3, 2]. Cependant ces algorithmes n'ont encore jamais été définis pour être utilisés directement dans des images en niveaux de gris. Nous proposons ici une solution pour étendre la reconnaissance en utilisant les informations des niveaux de gris d'une image. Bien qu'initialement conçu pour un outil semi-automatique de sélection de droites dans une application de modélisation 3d interactive, l'approche proposée répond aussi à des besoins plus généraux liés à l'extraction de paramètres.

1 Introduction

La reconnaissance de segments de droites discrètes est largement utilisée pour extraire des paramètres géométriques sur des objets discrets déjà segmentés, comme par exemple le périmètre, les tangentes, la courbure [13, 6, 9] ou l'approximation polygonale [1]. Différentes formulations ont été proposées pour cette reconnaissance (voir [8] pour un état de l'art). Certaines d'entre elles sont basées sur la définition arithmétique de segments de droites discrètes et ne sont pas toujours bien adaptées aux contours bruités.

Pour surmonter cette limitation, Debled-Rennesson et al. [3] et Buzer [2] ont proposé deux algorithmes équivalents pour reconnaître des segments flous, comportant un paramètre ν associé à la largeur. Ces algorithmes sont à la base de méthodes robustes au bruit. Par exemple l'estimateur de courbure proposé par Kerautret et Lachaud [7] délivre des valeurs précises mêmes sur des contours fortement bruités. La reconnaissance des segments flous a été à l'origine conçue pour des images binaires où plus exactement les données de départ de l'algorithme sont simplement la liste ordonnée des coordonnées des pixels. Cette caractéristique en restreint l'utilisation pour beaucoup d'applications d'analyse d'images et nous proposons ici d'étendre l'algorithme de reconnaissance aux images en niveaux de gris. L'idée principale est d'analyser directement les pixels à l'intérieur de larges bandes définies à partir d'une direction initiale. Des critères géométriques sont proposés pour sélectionner et classer des droites candidates selon l'information en niveaux de gris trouvée.

Cette extension est d'abord destinée à un outil semi-automatique de sélection de segments droits pour une application de modélisation 3D interactive à partir d'images issues de caméras. Dans ce contexte, une coopération étroite entre l'homme et la machine est encore incontournable pour sélectionner des indices visuels pertinents [10]. Les droites extraites sont essentiellement utilisées pour définir des orientations dans l'espace pour des tâches d'étalonnage de caméra ou de mise en correspondance d'objets 3D sur les images [4]. Compte tenu du grand nombre de segments à extraire au cours d'une session de modélisation, même des améliorations modestes de cette tâche élémentaire peuvent contribuer à une réduction notable de la durée d'une session complète de reconstruction 3D [5].

Dans la section suivante nous rappellons la définition des segments flous introduite par Debled-Rennesson et al. Ensuite nous decrivons notre approche pour étendre la reconnaissance de segments de droites discrètes aux images en niveaux de gris. Dans la section 4 nous présentons un processus interactif pour l'extraction de droites. Enfin, différentes remarques et perspectives seront présentées en conclusion.

2 Reconnaissance de segments flous

La notion de segment flou s'appuie sur la définition arithmétique de droites discrètes. Afin de la décrire brièvement, nous rappellons quelques définitions introduites dans [3]. Un ensemble de points entiers $\mathcal{S}$ appartient à une droite discrète $\mathcal{L}(a, b, \mu, \omega)$ si et seulement si tous les points vérifient $\mu \leq ax - by < \mu + \omega$. Les droites réelles d'équation $ax - by = \mu$ et $ax - by = \mu + \omega - 1$ sont définies comme les droites d'appuis supérieure et inférieure. Si $\mathcal{S}_b$ est un ensemble de points 8-connexe, une droite discrète $\mathcal{L}(a, b, \mu, \omega)$ est dite englobante pour $\mathcal{S}_b$ si tous les points de $\mathcal{S}_b$ appartiennent à $\mathcal{L}$. Enfin une droite englobante est dite optimale si sa distance verticale $\frac{\omega-1}{max(|a|,|b|)}$ est minimale, i.e si sa distance verticale est égale à la distance verticale de l'enveloppe convexe de l'ensemble S_b.

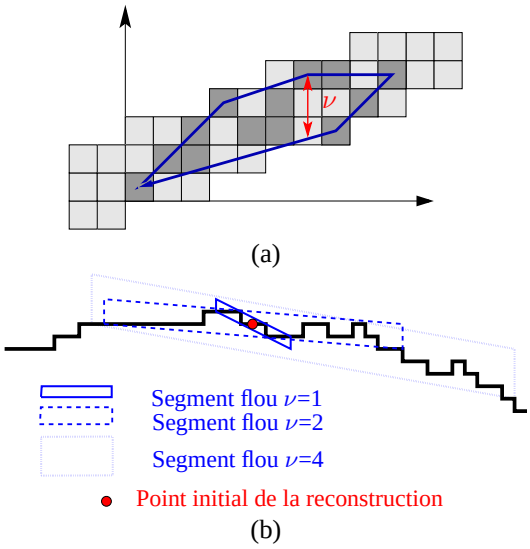


FIG. 1 – (a) Illustration d'un segment flou avec son enveloppe convexe et sa droite englobante (gris clair) définis sur un ensemble de pixels initiaux (gris foncé). (b) Illustration de la reconnaissance obtenue avec différentes valeurs du paramètre ν.

Definition 1 [3] Un ensemble de points $\mathcal{S}_b$ est un segment flou d'épaisseur ν si et seulement si sa droite englobante optimale a une épaisseur verticale inférieure ou égale à ν.

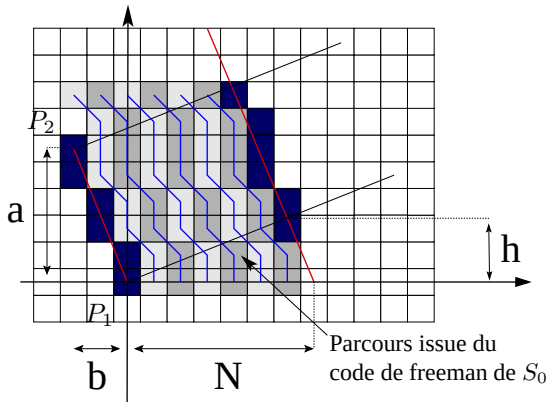
A partir de cette définition, les auteurs proposent un algorithme en temps linéaire pour reconnaître un segment flou à partir d'un ensemble de points discrets ordonnés. L'algorithme est basé sur le calcul de l'enveloppe convexe. Plus de détails sont donnés dans [3] ou [12] avec notamment différentes hypothèses sur les points discrets initiaux de l'algorithme. Un segment flou et son enveloppe convexe associée sont illustrés sur la figure Fig. 1 (a). Les pixels en gris foncé représentent les données initiales tandis que ceux en gris clair représentent la droite englobante optimale.
La figure Fig. 1 (b) illustre la reconnaissance des segments flous obtenus avec différentes valeurs du paramètre ν. La reconnaissance a été définie à partir des segments flous maximaux qui ne peuvent être étendus ni vers la gauche ni vers la droite. Comme on peut l'observer sur la figure, l'utilisation d'une grande valeur pour l'épaisseur ν permet de reconnaître des segments affectés de faibles variations.

3 Extension aux images en niveaux de gris

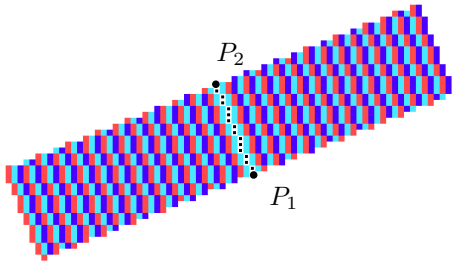
Afin d'étendre cette reconnaissance de segments flous aux images en niveaux de gris, nous effectuons un parcours directionnel dans l'image basé sur trois phases incrémentales d'affinement. La première consiste en une reconnaissance préliminaire afin d'obtenir une approximation de la direction $\mathfrak{D}^{(1)}$ associée à une structure linéaire proche de la direction initiale $\mathfrak{D}^{(0)}$. La seconde (resp. troisième) phases utilise la direction $\mathfrak{D}^{(1)}$ (resp. $\mathfrak{D}^{(2)}$) pour améliorer le processus de reconnaissance.
La direction $\mathfrak{D}^{(0)}$ est définie par deux points P_1 et P_2 placés de part et d'autre de la droite à reconnaître. Ces deux points définissent un segment de recherche initial $S_0^{(0)}$ associé à la direction initiale $\mathfrak{D}^{(0)}$. Les points candidats pour la reconnaissance de la droite sont extraits le long des segments de recherche. Il s'agit ensuite d'analyser successivement des segments de recherche de chaque coté de $S_0^{(0)}$ pour obtenir les points candidats pour l'algorithme de reconnaissance. L'extension de l'algorithme de reconnaissance repose sur le parcours des segments de recherche et sur l'analyse du gradient de l'image pour tenir compte de la cohérence spatiale dans l'image.
Une fois détecté un premier segment flou, une nouvelle direction de recherche $\mathfrak{D}^{(1)}$ est définie à partir de la direction orthogonale au segment détecté. Cette direction est alors utilisée pour définir un nouveau parcours directionnel. Le processus est itéré une dernière fois avec la direction $\mathfrak{D}^{(2)}$.
Dans la section suivante, nous détaillons d'abord le calcul des segments de recherche pour le parcours directionel, puis nous présentons la stratégie de sélection et de tri des pixels candidats, et enfin l'algorithme de reconnaissance.

3.1 Recherche directionnelle des segments

Un segment de recherche initial S_0 est défini par deux points $P_1 = (x_0, y_0)$ et $P_2 = (x'_0, y'_0)$ placés de part et d'autre du contour droit à détecter. L'image est analysée dans une direction orthogonale à S_0 le long de segments S_i parallèles à S_0, formant la région de recherche. La droite discrète contenant S_0 est donnée par $\mathcal{L}_0(a, b, \mu_0, \omega)$, où $b = x'_0 - x_0$, $a = y'_0 - y_0$ et $\omega = max(|a|, |b|)$.
Pour garantir que tous les pixels de la région de recherche sont parcourus une fois et une fois seulement, chaque segment de recherche S_i est inscrit dans une droite discrète naïve (8-connexe), obtenue par un décalage horizontal ou vertical d'un pixel à partir de la droite précédente (Fig. 2 (a)). Le segment initial S_0 est calculé par un algorithme arithmétique de discrétisation, dans lequel chaquel pixel est associé à une valeur de reste $r_i = r_0 + i \cdot min(|a|, |b|) \mod (max(|a|, |b|))$, $i \in [0, max(|a|, |b|)]$, avec une valeur $r_0 = 0$ pour le premier pixel P_1. Le segment S_i peut être obtenu directement à partir du segment initial en appliquant l'algorithme de numérisation avec des valeurs adéquates pour les coordonnées du pixel de départ et de son reste associé, ou bien incrémentalement en appliquant une permutation circulaire au code de Freeman associé au segment S_{i-1} (Fig. 2 (a)).
Ainsi, dans la cas d'une direction de recherche inscrite dans le premier octant (i.e. $a > -b > 0$), le segment d'ordre N est inscrit dans la droite discrète $\mathcal{L}_N(a, b, \mu_N, \omega)$, où $\mu_N = \mu_0 + N \cdot a$. Les coordonnées du pixel de départ sont $x_N = x_1 + N - \lfloor (N \cdot |b|)/a \rfloor$ et $y_N = y_1 + H$, où H est l'entier le plus proche de $h = \frac{a \cdot b}{a^2 + b^2}$. Le reste associé est $r_N = (N \cdot |b|) \mod a$. Des traitements analogues sont appliqués pour les autres



(a)



(b)

FIG. 2 – Calcul du N^{ieme} segment S_N (a) et exemple de région de recherche avec ses segments successifs (b).

octants.

3.2 Sélection des candidats dans les segments de recherche

Il existe différents moyens de sélectionner les pixels candidats pour le processus de reconnaissance. Comme nous cherchons à extraire des segments de droites contenus dans une image en niveaux de gris, il semble naturel d'utiliser les informations du gradient de l'image. Une autre alternative serait de comparer les profils d'intensité de l'image des candidats avec le profil au point initial dans la direction $\mathfrak{D}^{(0)}$. Dans ce travail, nous avons choisi d'explorer la première solution et nous laissons la deuxième comme perspective.

Pour la suite nous noterons par $g_x(p)$ et $g_y(p)$ les deux composantes du vecteur gradient $g(p)$ calculé dans l'image et $\|g(p)\|$ représentera l'amplitude $\sqrt{g_x^2(p)+g_y^2(p)}$. Chaque composante est calculée par un simple filtre Sobel avec une taille de voisinage fixée. La taille du filtre peut être ajustée au niveau de bruit présent dans l'image. Pour tous les tests de cet article, une taille standard de 3×3 s'est avérée suffisante.

Pour la suite, nous noterons par p_m le point d'ordre m d'un segment S_k. La première sélection des points candidats est définie en sélectionnant seulement le point p_m pour lequel la norme du gradient $\|g(p_m)\|$ est un maximum local. Afin de ne retenir que les points significatifs nous fixons une déviation angulaire maximale θ_{max} autorisée entre la direction $\mathfrak{D}$ et la direction du vecteur gradient. Ainsi pour une phase i de l'algorithme, nous enlevons de la liste des candidats tous les points pour lesquels la déviation angulaire est plus grande que $\theta_{max}^{(i)}$. Plus formellement la pré-sélection $\mathcal{C'}_k^i$ pour une phase i est définie de la façon suivante :

$$\mathcal{C'}_k^i = \Big\{ p_m \in S_k | (\ \exists\ a,b\ ;\ \forall j \in [a,b],$$
$$\|g(p_j)\| \leq \|g(p_m)\|) \wedge (|U_{\mathfrak{D}^{(i)}} \cdot U_{g(p_m)}| > \cos(\theta_{max}^{(i)})) \Big\}\ ;$$

où $U_{\mathfrak{D}^{(i)}}$ et $U_{g(p_m)}$ sont respectivement les vecteurs unité associés aux directions $\mathfrak{D}^{(i)}$ et $g(p_m)$ avec $i \in \{0,1,2\}$. La liste finale $\mathcal{C}_k^i$ est obtenue après un tri décroissant en fonction des valeurs $\|g(p_m)\|$. Comme dans la première phase de l'algorithme la direction initiale n'est pas assez précise nous relâchons la contrainte en fixant la valeur de $\theta_{max}^{(0)}$ à $\frac{\pi}{2}$. Pour la deuxième et troisième phase le paramètre θ_{max} est réglé à $\theta_{max}^{(1)} = \frac{\pi}{8}$ et $\theta_{max}^{(2)} = \frac{\pi}{10}$. Ces valeurs empiriques ont été validées sur des images réelles. Cependant ce paramètre peut être réglé de façon à adapter la convergence de l'algorithme au contexte applicatif et à la nature des images.

3.3 Algorithme de reconnaissance

L'algorithme de détection de segment droit s'appuie sur deux primitives de l'algorithme de reconnaissance de segment flou :

- `SegmentFlou(`P_0, P_1,P_2,ν`)` qui délivre un segment flou initial de largeur ν à partir de trois points,
- `ajouter(`P_i`)` qui tente d'étendre le segment flou avec un nouveau point.

L'algorithme de détection s'appuie sur la définition de la liste de candidats fournie précédemment. Les entrées sont les deux listes L_R et L_L de candidats $\mathcal{C}_k$ trouvés pour chaque segment de recherche S_k, et la liste de candidats $\mathcal{C}_0$ correspondant au segment initial S_0. Comme une liste de candidats $\mathcal{C}_k$ peut être vide, pour garantir l'initialisation de l'algorithme, les listes de candidats sont décalées jusqu'à ce qu'une liste non vide soit trouvée pour $\mathcal{C}_1$ et $\mathcal{C}_0$. Le premier paramètre de l'algorithme 1 est la largeur ν du segment flou détecté. Il est ajusté en tenant compte de la précision escomptée du segment droit à extraire. Une petite valeur de ν fournit un segment fin, mais sa reconnaissance est sensible aux petites variations. En revanche, avec une valeur plus forte, le bruit est mieux pris en compte. Un deuxième paramètre, τ, contrôle la tolérence aux interruptions lors de la détection. Il fournit le nombre de listes de candidats vides successives tolérées avant d'arrêter l'algorithme 1. Il fixe la longueur des plages de non détection du segment. Cela renforce la robustesse au bruit et permet de franchir des zones d'occultation du segment.

L'algorithme 2 sert à construire le segment initial à partir des listes de candidats $\mathcal{C}_0$, $\mathcal{C}_1^R$ et $\mathcal{C}_1^L$ et d'un troisième paramètre contrôlant le nombre maximal de segments droits proposés. Ce paramètre est particulièrement

Algorithme 1 : Algorithme de reconnaissance

Entrée : $\mathcal{C}_0 :=$ la liste des candidats du scan central S_0.
$L_R := \{\mathcal{C}_1^R, \ldots, \mathcal{C}_k^R, \ldots, \mathcal{C}_M^R\}$ l'ensemble de listes des candidats $\mathcal{C}_k^R$ pour un scan à droite S_k^R ;
$L_L := \{\mathcal{C}_1^L, \ldots \mathcal{C}_k^L, \ldots, \mathcal{C}_N^L\}$ l'ensemble de listes des candidats $\mathcal{C}_k^L$ pour un scan à gauche S_k^L;
Paramètres : réel ν ; entiers τ, numSolution, phase;
Résultat : Segment Flou Maximal
SegmentFlou := initialiseSegmentFlou($\mathcal{C}_0$, $\mathcal{C}_1^L$, $\mathcal{C}_1^R$, ν , numSolution, phase);
entier nbInteruptGauche, nbInteruptDroite :=0 ;
booléen scanDroitStoppé, scanGaucheStoppé :=faux ;
entier k :=1;
tant que *NON scanDroitStoppé OU NON scanGaucheStoppé* **faire**
- **si** *NON scanDroitStoppé* **alors**
  - pointAjouté :=faux;
  - **tant que** $\mathcal{C}_k^R$ *n'est pas vide ET NON pointAjouté* **faire**
    - p :=premier élément de $\mathcal{C}_k^R$,enlever p de $\mathcal{C}_k^R$;
    - pointAjouté :=SegmentFlou.ajouter(p) ;
  - **si** *pointAjouté* **alors**
    - nbInteruptDroite :=0;
  - **sinon**
    - nbInteruptDroite=nbInteruptDroite+1;
- **si** *NON scanGaucheStoppé* **alors**
  - pointAjouté :=faux;
  - **tant que** $\mathcal{C}_k^L$ *n'est pas vide ET non pointAjouté* **faire**
    - p := premier élément de $\mathcal{C}_k^L$, enlever p de $\mathcal{C}_k^L$;
    - pointAjouté :=SegmentFlou.ajouter(p) ;
  - **si** *pointAjouté* **alors**
    - nbInteruptGauche :=0;
  - **sinon**
    - nbInteruptGauche :=nbInteruptGauche+1;
- scanDroitStoppé :=(nbInteruptDroite=τ OU k=M);
- scanGaucheStoppé :=(nbInteruptGauche=τ OU k=N);
- incrementer k ;

Algorithme 2 : Construction du segment flou initial : initialiseSegmentFlou

Entrée : liste de candidats $\mathcal{C}_0$, $\mathcal{C}_1^L$, $\mathcal{C}_1^R$; réel ν ; entier numSolution, phase ;
Résultat : SegmentFlou
si *phase=0* **alors**
- P_c := le numSolution^{ieme} élément de $\mathcal{C}_0$;
- P_1^R := le point le plus proche de P_c extrait de tous les éléments de $\mathcal{C}_1^R$;
- P_1^L := le point le plus proche de P_c extrait de tous les éléments de $\mathcal{C}_1^L$;

sinon
- P_c := le premier élément de $\mathcal{C}_0$;
- P_1^R := le premier élément de $\mathcal{C}_1^R$;
- P_1^L := le premier élément de $\mathcal{C}_1^L$;

retourner SegmentFlou(P_c, P_1^R, P_1^L, ν) ;

utile quand le segment initial P_1P_2 coupe plusieurs segments dans l'image. Cette sélection s'applique uniquement à la première phase du processus de détection, car pour les deuxième et troisième phase, nous disposons déjà d'une bonne estimation du segment flou.
L'algorithme de détection se résume dans les six étapes suivantes :

1. Précalcul du gradient d'image $g_x(p)$ et $g_y(p)$ pour tous les pixels p.
2. Calcul de toutes les listes de candidats L_R et L_L en partant des deux points P_1 et P_2.
3. Recherche d'un segment flou SF par l'algorithme 1 en mettant la variable $phase$ à 0.
4. Déplacement de P_1 et P_2 de sorte que $\|P_1P_2\| = 2\nu$, que le centre de P_1P_2 coïncide avec le premier point de SF et que la direction P_1P_2 soit orthogonale à la direction de SF.
5. Relance des étapes 2., 3. et 4. avec la variable $phase$ mise à 1.
6. Relance des étapes 2., 3. et 4. avec la variable $phase$ mise à 2.

Les validations expérimentales indiquent que les deux relances des phases 5. et 6. sont suffisantes, aucune amélioration tangible n'ayant pu être observée en augmentant ce nombre d'itérations. L'affinement progressif de la solution au fil des itérations est illustré en Fig. 5.

4 Extraction semi-automatique de segments droits

Pour extraire interactivement un segment dans une image, on pourrait se contenter de spécifier un point proche pour définir une zone de recherche initiale. Un segment pourrait alors être trouvé en s'appuyant sur une analyse locale du gradient. Mais ce mode opératoire ne fournit pas d'information valide pour orienter le processus de reconnaissance, ce qui le rend très sensible à la présence d'autres segments à proximité du point initial. Il est préférable de conserver

le contrôle de la direction initiale.

S'il existe de nombreuses publications traitant de la détection interactive de contours, il est plus difficile d'en trouver quand on se restreint à l'extraction interactive de contours droits. Rodrigo *et al.* [11] ont proposé une approche basée sur une technique de minimisation d'energie pour affiner et étendre un segment initial, mais la solution initiale fournie par l'opérateur doit être assez proche du segment à détecter et les temps de traitement sont relativement longs pour converger vers une solution satisfaisante. Even et Malavaud [5] ont présenté une approche séquentielle dans laquelle un segment initial tracé grossierement par l'opérateur est attiré vers la meilleure solution proposée par un traitement automatique. Ce traitement s'appuie sur l'enchainement de deux phases de transformée de Hough appliquées aux points de contours proches du segment initial, en utilisant une première grille de faible résolution puis une grille plus fine pour affiner la solution. Des temps d'extraction beaucoup plus court ont ainsi pu être obtenus par rapport à une extraction manuelle, mais là aussi, la solution fournie par l'opérateur doit être assez proche du segment à extraire. Nous souhaitons ici relâcher encore plus cette contrainte.

Dans notre approche, l'opérateur se contente de barrer le segment qu'il désire extraire dans l'image, sans contrainte forte sur la direction fournie. Il suffit que les extrémités de la barre se situent de part et d'autre du segment. Elles sont immédiatement utilisées pour définir le segment de recherche initial P_1P_2 et lancer l'algorithme d'extraction de segment flou. Cette technique s'avère plus souple pour l'opérateur et plus rapide.

Plusieurs expérimentations sont présentées sur la figure Fig. 3. Pour toutes les images (sauf (b)) des paramètres identiques ont été utilisés pour la détection de droites. Les lignes rouges représentent la direction donnée par l'utilisateur tandis que celles en bleu représentent le résultat de la détection obtenue avec l'algorithme. L'image (c) illustre la possibilité de sélectionner des solutions multiples quand la ligne définie par l'utilisateur coupe plusieurs contours. Les images (e-g) montrent la capacité de l'algorithme pour reconnaître des zones définies par un très faible gradient. Une croix a été dessinée manuellement sur l'image (e) de la figure Fig. 3 avec un niveau de gris très proche du niveau moyen du voisinage (zone coloriée en bleu sur l'image (f)). Malgré la faible valeur du gradient, la croix est bien détectée (image (g)).

La figure Fig. 4 illustre la reconnaissance obtenue avec plusieurs valeurs de ν sur une zone courbée. On peut observer qu'une petite valeur de ν provoque une détection précise sur une zone limitée tandis qu'avec un grande valeur de ν nous obtenons une détection plus grossière de la zone.

Pour montrer le comportement du processus de reconnaissance sur des données bruitées, nous avons appliqué l'algorithme sur une version bruitée de l'image de la figure Fig. 3 (a) obtenue à partir d'un bruit gaussien avec un écart type $\sigma = 10$. La figure Fig. 5 montre les trois étapes du processus de reconnaissance obtenues sur deux zones de l'image (colonne de gauche et de droite). Les paramètres de l'algorithme ($\nu = 5$ et $\tau = 5$) n'ont pas été changé et la taille du masque pour calculer le gradient a été laissé à 3×3. Pour le premier exemple (a-c) on peut voir qu'à l'étape (3) la reconnaissance est bien stoppée (grâce à la valeur de $\theta^{(3)}_{max}$). Le second exemple (d-f) montre plus de sensibilité au bruit si on le compare aux résultats précédents de la figure Fig. 3 (a). Plus généralement, la présence du bruit procure plus de sensibilité à la direction initiale et réduit le nombre de segments détectés. L'utilisation d'une taille de masque plus grande apparaît nécessaire pour traiter les images particulièrement bruitées.

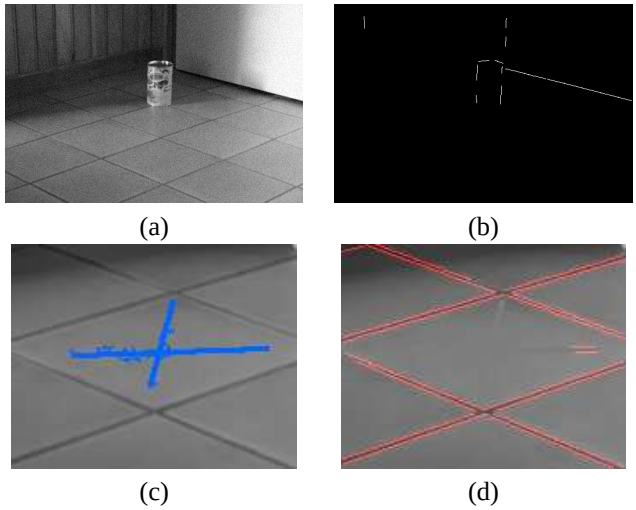
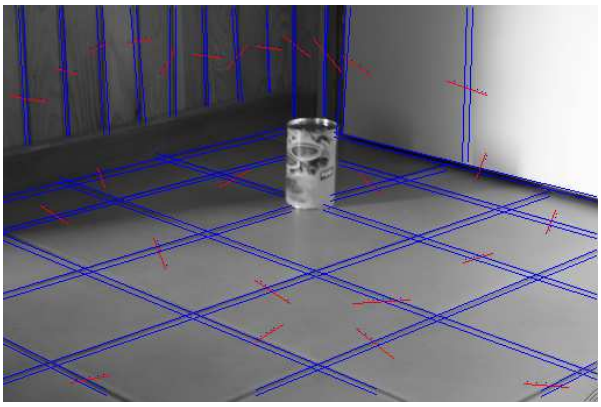


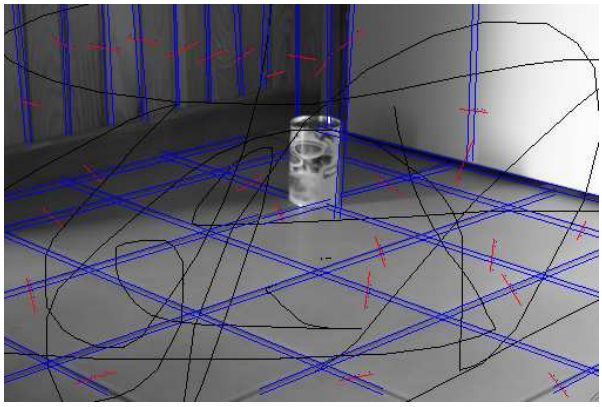
FIG. 6 – Application de la méthode LSD [15] sur les images bruitées de la figure Fig. 5 (image (a)) et sur l'image de la figure Fig. 3(e). Les résultats présentés par les images (b) et (d) ont été obtenus par l'algorithme LSD à partir de la page de démonstrations des auteurs [14].

Une vidéo présentant la reconnaissance peut être téléchargée à l'adresse suivante :
`http://www.loria.fr/~kerautre/LiveBS`

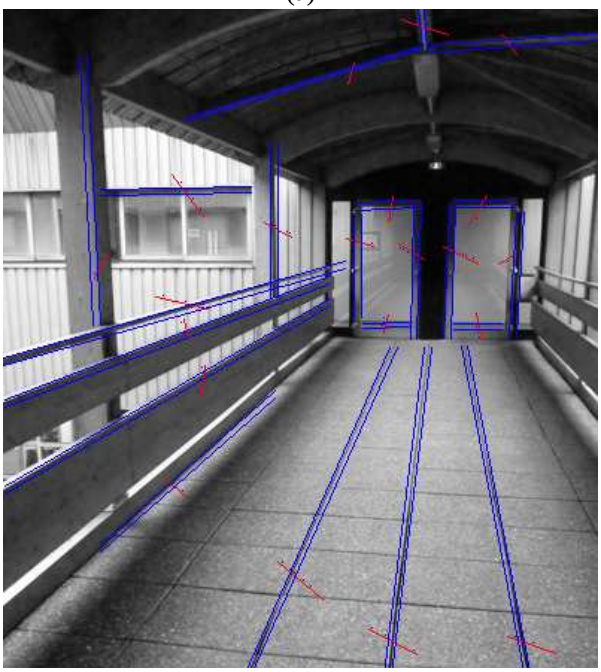
Autres expérimentations. Pour mesurer la performance de l'approche proposée nous avons effectué des expérimentations à partir de la méthode appelée *LSD* récemment introduite par Von Gioi et al. [15]. Cette approche a l'avantage d'avoir aucun paramètre et peut être calculée en temps linéaire. La figure Fig. 6 montre les résultats obtenus sur l'image bruitée de la figure Fig. 5. Il est possible de voir que la méthode est sensible à la présence de bruit car peu de segments ont été détectés. Une autre expérimentation a été faite sur l'image présentant une zone de très faible gradient (figure Fig. 3(e)). On peut observer qu'un seul petit morceau de segment de droite a été détecté. Toutes ces expérimentations ont été effectuées à partir de la page internet de démonstration des auteurs [14] et les résultats sont accessibles sur la page d'archive.



(a)

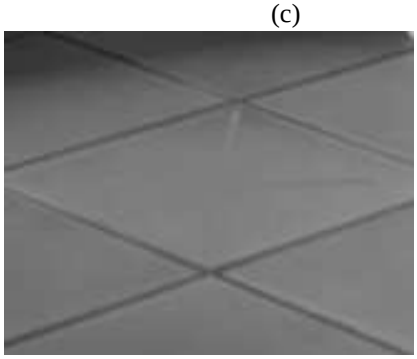


(b)

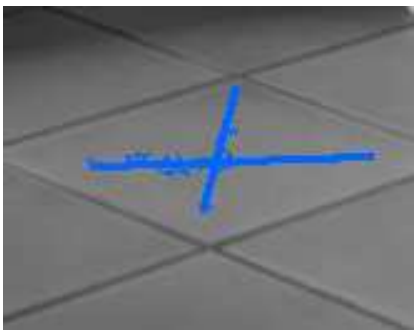


(c)

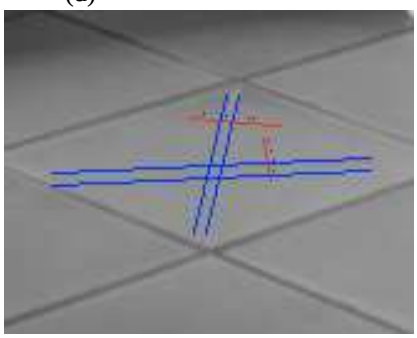
(d)



(e)

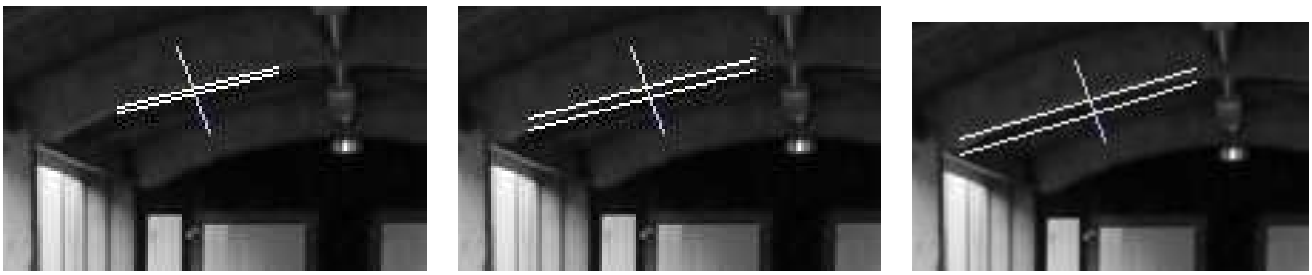


(f)



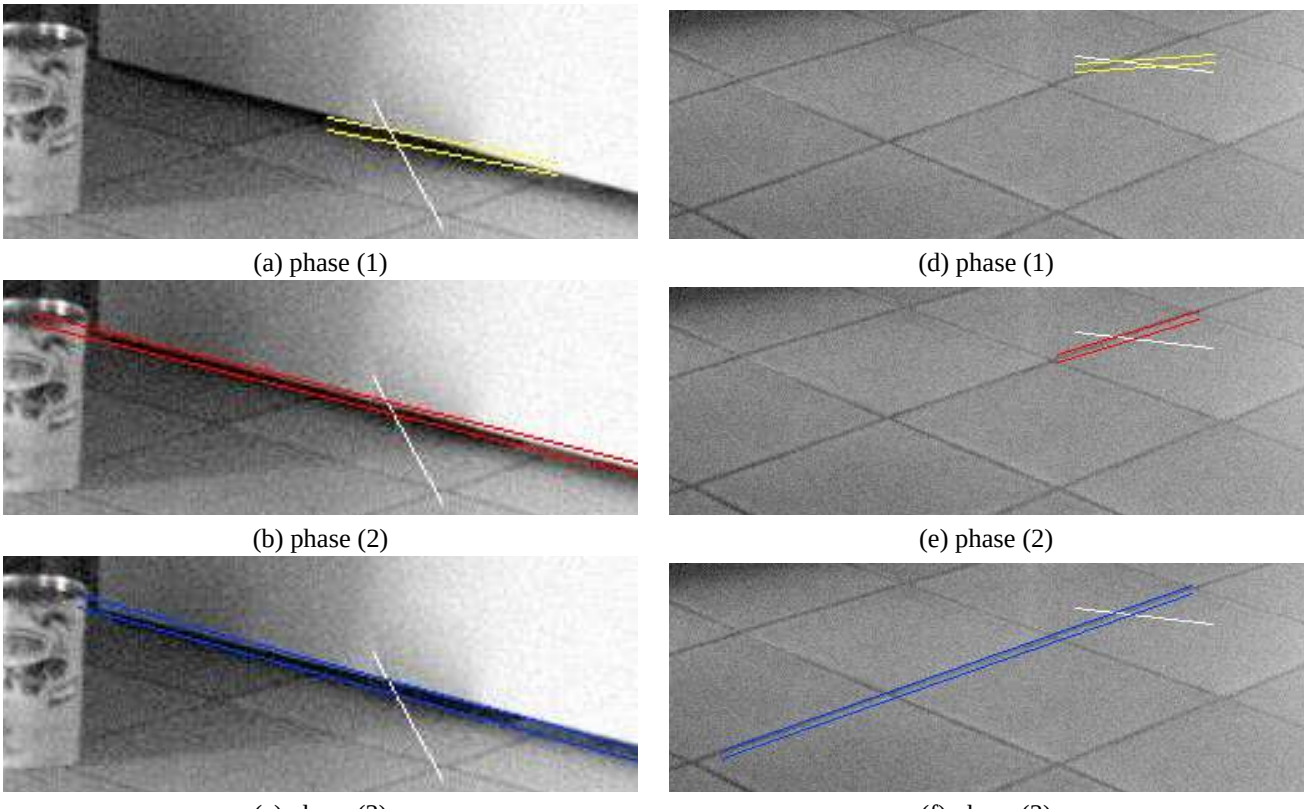
(g)

FIG. 3 – Résultats obtenus avec les mêmes paramètres : $(\nu = 5, \tau = 5)$ hormis pour l'image (b) $(\nu = 5, \tau = 8)$ qui illustre la détection obtenue que sur une image manuellement dégradée. L'image (c) illustre la détection sur une autre photographie tandis que l'image (d) montre la sélection de plusieurs segments coupés par le segment initial. Les images (e-g) illustrent les performances obtenues avec de faibles valeurs du gradient.



(a) $\nu = 2$ (b) $\nu = 4$ (c) $\nu = 7$

FIG. 4 – Illustration de la reconnaissance obtenue avec différentes valeurs du paramètre ν. Le segment obtenu avec une faible épaisseur montre une détection précise mais peu étendue (a) tandis que la taille de la détection augmente progressivement avec l'augmentation de l'épaisseur sur les images (b) et (c).



(a) phase (1) (d) phase (1)

(b) phase (2) (e) phase (2)

(c) phase (3) (f) phase (3)

FIG. 5 – Illustration des trois phases de l'algorithme de reconnaissance appliqué sur une version bruitée de l'image de la figure Fig. 3(a). L'image source a été obtenue avec un bruit gaussien d'écart type $\sigma = 10$. Les paramètres n'ont pas été modifiés sur cet exemple ($\nu = 5$ et $\tau = 5$). Chaque phase est illustrée en affichant les segments y sont reconnus (représenté par une couleur distincte).

FIG. 7 – Illustration de la perspective d'extraction de contours par suivi des segments maximaux.

5 Remarques et perspectives

Nous avons proposé une méthode simple et efficace pour étendre la reconnaissance des segments flous dans des images en niveaux de gris. Cette première extension permet d'effectuer une détection rapide, à la volée, et a besoin seulement de peu de paramètres avec une interprétation géométrique simple (l'épaisseur ν du segment flou et τ pour le seuil de la longueur de l'interruption). La prochaine étape de ce travail est d'adapter la reconnaissance dans une approche multi-resolution de façon à être capable de détecter des segments définis à travers de nombreuses variations comme illustré sur la figure ci dessus. De plus nous envisageons de mener des campagnes d'évaluation ergonomique pour apprécier plus finement les apports du mode semi-automatique proposé pour extraire des segments droits dans une image. Enfin comme autre perspective il serait intéressant d'étendre l'approche de façon à extraire un contour de l'image qui ne soit pas un segment. Une première approche possible est de continuer une nouvelle reconnaissance à la fin du premier segment maximal et de recommencer une nouvelle reconnaissance en rajoutant des contraintes géométriques. Cette idée est illustrée sur la figure Fig. 7.

Références

[1] P. Bhowmick and Bhargab B. Bhattacharya. Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Trans. on PAMI*, 29(9), 2007.

[2] L. Buzer. A simple algorithm for digital line recognition in the general case. *Pattern Recognition*, 40(6) :1675–1684, 2007.

[3] I. Debled-Rennesson, F. Feschet, and J Rouyer-Degli. Optimal blurred segments decomposition of noisy shapes in linear times. *Comp. & Graphics*, 30 :30–36, 2006.

[4] P. Even. A generic procedure for interactive 3D model registration on images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B5) :204–209, 2004.

[5] P. Even and A. Malavaud. Semi-automated edge segment specification for an interactive modelling system of robot environments. *International Archives of Photogrammetry and Remote Sensing*, 33(B5) :222–229, 2000.

[6] F. Feschet and L. Tougne. Optimal time computation of the tangent of a discrete curve : Application to the curvature. In *Proc. of Int. Conf. on DGCI*, volume 1568 of *LNCS*, pages 31–40. Springer, 1999.

[7] B. Kerautret and J.-O Lachaud. Curvature estimation along noisy digital contours by approximate global optimization. *Pattern Recognition*, 42(10) :2265–2278, October 2009.

[8] R. Klette and A. Rosenfeld. Digital straightness–a review. *Discrete Applied Mathematics*, 139(1-3) :197 – 230, 2004.

[9] J.-O. Lachaud, A. Vialard, and F. de Vieilleville. Fast, accurate and convergent tangent estimation on digital contours. *Image and Vision Computing*, 25 :1572–1587, 2007.

[10] T. Ohlhof, E. Gülch, H. Müller, C. Wiedemann, and M. Torre. Semi-automatic extraction of line and area features from aerial and satellite images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(B3) :471–476, 2004.

[11] R. Rodrigo, W. Shi, and J. Samarabandu. Energy based line detection. In *Proc. of Canadian Conf. on Electrical and Comp. Engineering*, pages 2061–2064. IEEE, 2006.

[12] T. Roussillon, L. Tougne, and I. Sivignon. Computation of binary objects sides number using discrete geometry application to automatic pebbles shape analysis. In *Proc. 14th International Conference on Image Analysis and Processing (ICIAP)*, pages 763–768, september 2007.

[13] A. Vialard. Geometrical parameters extraction from discrete paths. In *Proc. of Int. Conf. on DGCI*, volume 1176 of *LNCS*, pages 24–35. Springer, 1996.

[14] R. G. von Gioi, J. Jakubowicz, J-M. Morel, and G. Randall. Lsd demo disponible en ligne : `http://mw.cmla.ens-cachan.fr/megawave/demo/lsd/`.

[15] R. G. von Gioi, J. Jakubowicz, J-M. Morel, and G. Randall. Lsd : A fast line segment detector with a false detection control. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 2008. preprint.