PREMIER COLLOQUE IMAGE

Traitement, Synthèse, Technologie et Applications

BIARRITZ — Mai 1984 —

DE L'EXTRACTION DE SEGMENTS DE DROITES D'UNE SEQUENCE DE POINTS

EXTRACTING STRAIGHT LINE SEGMENTS FROM A POINT SEQUENCE

Augustin LUX

LIFIA / IMAG - BP 68 - 38402 ST MARTIN-D'HERES CEDEX

RESUME

Nous analysons le problème algorithmique suivant: Etant donné une séquence de points, formant une ligne digitale, trouver une séquence de segments de droites approchant "au mieux" la séquence de points. Nous présentons plusieurs algorithmes, analysant les propriétés suivantes: Précision des résultats, indépendance du choix du point initial, caractère incremental, place mémoire, temps d'exécution. Les algorithmes connus ont tous des défauts majeurs par rapport à ces critères. Nous proposons un nouvel algorithme combinant plusieurs avantages: Place mémoire constante, temps linéaire, et approximation aux moindres carrés.

SUMMARY

We analyze the following algorithmic problem: Given a sequence of points defining a digital line, determine the sequence of straight line segments present in the point sequence. We present several algorithms for which we analyze the following properties: Precision of results, independence of the choice of initial points, noise independence, incrementality, memory space, execution time. All published algorithms have major difficulties with some of these criteria. We propose a new algorithm realizing an interesting combination of useful properties - constant memory requirement, linear time, and a least squares fit.

DE L'EXTRACTION DE SEGMENTS DE DROITES D'UNE SEQUENCE DE POINTS

EXTRACTING STRAIGHT LINE SEGMENTS FROM A POINT SEQUENCE

Augustin LUX

I Introduction

Dans cette communication nous étudions un problème des plus simples en reconnaissance de formes: La recherche de segments de droite dans une image digitale.
Ce problème est d'une importance pratique considérable:

* De nombreux systèmes de reconnaissance d'objets utilisent les segments de droite comme éléments de base de description d'objets, et certains les utilisent exclusivement.
* Le segments de droite permettent d'approcher d'autres types de courbes avec la précision voulue, quoique au prix d'un nombre elévé de segments.
* Comme nous le verrons, il existe des algorithmes efficaces et précis pour trouver les segments de droite, ce qui semble ne pas être le cas pour d'autres types de courbes. (Cf. par exemple le travail de [Agin] sur la recherche d'arcs élliptiques).

Toutefois, la simplicité du problème n'est qu'une apparence. Comme dans tant de problèmes de reconnaissance de formes, l'analyse détaillée met en évidence des contraintes contradictoires qu'on souhaiterait voir satisfaites par une solution.

Nous énonçons le problème d'abord sous la forme suivante: Etant donné une ligne digitalisée sous forme d'une séquence de n points (P1 ... Pn), déterminer les segments de droite dans cette ligne. Les points Pi font partie d'une image à deux dimensions et sont définis par deux coordonnées entières. Le terme "ligne digitalisée" signifie que chaque point est voisin immédiat du point précédent (et donc du point suivant). Afin de ne pas alourdir l'exposé, nous supposons que la ligne ne contient pas de boucle; Pi # Pj pour i # j, i,j = 1,... n.

Une première approche "naive" du problème permet d'en illustrer les pièges. On peut considérer les coordonnées des points comme des valeurs exactes et délimiter les droites coincidant parfaitement avec ces points. Cela donne un résultat satisfaisant dans le cas de la figure 1a. Cependant, en dehors de ce cas exceptionnel d'alignement de la ligne avec un des huit axes principaux de l'image, on arrive à une fragmentation excessive de la ligne (figure 1.b). Pourtant, c'est une solution formellement correcte au problème tel que nous venon de le poser. Nous avons seulement commis l'erreur de considérer les données comme exactes, negligeant le bruit de digitalisation. Une technique classique permet d'éviter cette difficulté - l'approximation aux moindres carrées. Cependant, tout nuage de points admet une droite l'approchant, mais cela n'est pas toujours la solution souhaitée (figure 1.c).
Ces réflexions montrent que l'énoncé intuitif du problème manque de précision. Il est nécessaire de mieux caractériser le résultat. Ceci sera fait implicitement par chaque algorithme. Nous retenons pour l'instant l'existence de deux sous-problèmes importants:

1/ La définition des "points de cassure".
2/ La détermination des équation des résultats (dans le plan réel).

II Découpage récursif

II.1 Description

Le découpage récursif est un algorithme élégant resolvant simultanement les deux sous-problèmes. Cet algorithme est utilisé dans de nombreuses applications [1,3,6]. Le principe est le suivant: Soit VP un vecteur contenant les points de la ligne à découper. On considère la droite passant par le premier et le dernier point; soit Pmax le point le plus éloigné de cette droite (fig. 2), à une distance Dx. Deux cas peuvent se présenter.

a/ Dx est inférieur à une certaine "tolérance latérale" EPSD. On considère alors que la ligne est composée d'un seul segment de droite.
b/ Dx est supérieur à EPSD. On considère alors Pmax comme un point de cassure. Il faut donc découper séparement la première partie de la ligne, allant jusqu'à Pmax, et la deuxième partie commençant à Pmax.

La terminaison des appels récursifs est garantie, car le nombre de points à traiter diminue strictement à chaque appel imbriqué. Par contre, si la valeur de EPSD est faible, cela peut amener à une forte fragmentation de la ligne, comme dans la figure 1.b. Pour cette raison il est intéressant d'introduire un deuxième paramètre, EPSL, correspondant à un nombre de points minimum par segment.

L'algoritmhe fournit comme résultat un vecteur de segments de droites VD, avec les propriétés suivantes:

1/ Chaque segment approche une partie (Pi ... Pj) de la ligne.
2/ Les parties correspondant à deux segments différents sont disjointes, mais la réunion de l'ensemble ne forme pas nécessairement la ligne entière,
3/ Les segments de droites du résultats sont obtenus dans le vecteur VD dans l'ordre de la ligne.

II.2 Evaluation

Grace au deux paramètres EPSD et EPSL il est possible d'adapter l'algorithme aux caractéristiques de la ligne analysée. Leur choix ne peut être arbitraire, comme le montre l'exemple de la figure 3. D'un point de vue pratique, cet algorithme donne des résultats très satisfaisants, même s'ils dépendent théoriquement du choix des points initiaux et qu'ils n'ont pas des propriétés mathématiques très poussées: Par construction, une bande de largeur 2*EPSD le long d'un segment contient les point (Pi Pi+1 ... Pj). Mais le segment n'est pas maximal - les points Pi-1 et Pj+1 peuvent faire partie de cette bande. D'autre part, les segments de droites entre les points de cassure ne correspondent pas à une approximation optimale. Il n'y a cependant aucun inconvenient à compléter le découpage par une approximation aux moindres carrés pour chaque segment (cf. IV).

Par une coincidence remarquable, le découpage recursif a la même structure algorithmique que le tri par segmentation dont on peut reprendre l'évaluation [12].

En ce qui concerne la place mémoire, il faut stocker l'ensemble des n points, et il faut une pile pour la gestion des appels récursifs. Dans le cas le plus

défavorable, la taille de la pile doit être de l'ordre du nombre de droites du résultat, c.a.d. de l'ordre de n/EPSL. (Il suffit d'empiler une seule valeur, celle de IX, indexe de Pmax dans la table).
Une légère variation de l'algorithme - l'inversion de l'ordre des appels récursifs si I2 - IX < IX - I1 - permet de reduire la taille de la pile à l'ordre

$$LOG_2(n/EPSL)$$

Il faut alors empiler deux valeurs par appel recursif, et les droites ne sont pas construites dans l'ordre d'apparition dans la ligne.

Pour le temps d'exécution, nous nous concentrons sur le nombre de calculs de distance entre un point et une droite. Ce nombre est de l'ordre n**2 dans le cas le plus défavorable, et n dans le cas le plus favorable. La valeur moyenne est fonction du type de la ligne découpée; on peut l'estimer à

$$n*LOG_2(n/EPSL).$$

III Reconnaissance syntaxique

III.1 Description

La méthode syntaxique exploite pour la reconnaissance un modèle génératif de droite digitale. Le modèle part d'une image idéalisée dans laquelle un pixel est représenté par un carré du plan. Un pixel de coordonnées entières (X,Y) couvre l'ensemble des points (x,y) à coordonnées réelles tels que

$$X <= x < X+1,\ Y <= y < Y+1$$

Nous considérons qu'un pixel fait partie de "l'image digitale" de la droite si et seulement si l'intersection entre le pixel et la droite est non-vide; de plus, on élimine les angles droits au profit de diagonales. Cette image constitue une ligne digitale. La figure 4 en montre un exemple, avec une droite passant par l'origine et de pente m faible - 0 < m < 1. Dans ce cas, la ligne est composée des pixels (X,Y) tels que

$$Y = \lfloor m*X \rfloor$$

On constate que dans ce cas la ligne est formé d'une suite de "barrettes", et que la longueur des barrettes varie au plus de 1. Cela s'explique facilement. La longueur l de la première barrette à partir de l'origine est telle que

$$l*m < l <= (l+1)*m \qquad (1)$$

c.a.d. l est le quotient entier l = 1 DIV m. Avec l'hypothèse de recurrence d'une ligne de n barrettes de longueur li

$$(2) \qquad \sum_{i=1}^{n} (li*m) = m* \sum_{i=1}^{n} li < n < m*(1+\sum_{i=1}^{n} li)$$

on additionne l'inégalité (1)

$$l*m+\sum (li*m) < n+1 <= m*(2+l+\sum li)$$

La deuxième inégalité implique ln = l ou ln = l+1.

Une étude plus détaillé (mais sans difficulté, étant donné les cas de symétrie) montre que les régularités rencontrées dans le cas 0 < m < 1 correspondent au cas général. Elles nous permettent de caractériser une droite digitale par la grammaire des codes direction (codes de Freeman):

Propriété. La suite S des codes direction d'une droite digitalisée est de la forme

$$S = \{ C_1^{\hat{l}\{+1\}} C_2 \}^+ \qquad (3)$$

Au plus deux codes différents figurent dans S. Ces codes correspondent à deux directions voisines, et un seul de ces codes peut se repéter. La longueur d'une repétition peut prendre deux valeurs différentes, soit $\hat{l}$ et $\hat{l}+1$, $\hat{l} = l-1$.

L'image d'un segment d'une droite est un mot de ce langage, sauf pour la première et la dernière barrette qui peuvent être tronquées.

Remarque: On peut caractériser S plus finement. Avec un raisonnement analogue au précédent on montre que les barrettes de longueur l et l+1 sont disposées en fonction du quotient

$$l_2 = l \text{ DIV } r$$

où r est le reste de la division l DIV m; et ainsi de suite. Seulement si m est rationnel on obtient effectivement un langage régulier. Cependant, le bruit dû au capteur rend illusoire l'exploitation pratique de cette structure précise.

La reconnaissance d'un segment de droite suivant ces principes constitue dont l'analyse d'un langage d'états finis dans les codes direction. L'algorithme comporte deux phases:

1/ Acquisition des paramètres.
Pour trouver C1 et C2 il suffit d'analyser une barrette plus un code direction supplémentaire. On vérifie alors que C1 et C2 sont des codes voisins. La longueur k de la barrette fournit la valeur de l ou l+1 (ce qui se précisera par la suite).
2/ Fonctionnement en reconnaissance suivant la formule (3).

Remarque: Il faut également tenir compte de deux cas particuliers. Un segment peut être formée d'une seule barrette; et toutes les barrettes d'un segment peuvent être d'une même longueur.

III.2 Evaluation

La reconnaissance syntaxique fonctionne comme un automate d'états finis avec une bande d'entrée contenant les points de la ligne, et une bande de sortie avec les segments du résultat. L'algorithme n'a pas besoin de stocker les point de la ligne dans un vecteur, et fournit les résultats au fur et à mesure du parcours de la ligne. Nous appellons ce caractère incrémental.
Un algorithme incrémental

- utilise une place mémoire constante
- a un temps d'exécution linéaire en n.

Il est donc à la fois économique et rapide.

L'inconvénient majeur de cet algorithme réside dans la sensibilité au bruit. En effet, les segments de droites rencontrés dans les images réelles ne correspondent pas toujours très bien au modèle théorique, et il est difficile de rendre l'algorithme

plus stable au bruit en élargissant les contraintes imposées par la syntaxe sans compromettre la précision des résultats.

IV Approximation aux moindres carrés

Dans l'algorithme de reconnaissance syntaxique on peut déterminer l'équation d'une droite directement. La formule (2) donne un encadrement pour la pente m, et la droite doit passer par le point qui est au centre de chaque barrette. En présence d'un bruit, ou dans le cas de l'algorithme par découpage, il est plus intéressant d'utiliser l'approximation aux moindres carrés. Les formules les plus connues concernent l'approximation d'une droite à partir de la forme

$$y=m*x + b$$

Dans notre cas, cette représentation est mal adaptée; elle n'est pas valable pour les droites verticales. De plus, l'erreur minimisée n'est pas la distance des points à la droite, mais la "distance verticale", ce qui peut donner des résultats paradoxaux. Il est préférable de représenter les droites par la forme normale

$$x*\cos(phi) + y*\sin(phi) + r = 0$$

La solution de ce problème d'approximation est donnée dans [4]. Par une translation on amène l'origine du système de coordonnées dans le centre des points Pi

$$P'_i = P_i - \frac{1}{n}\sum_{j=1}^{n} P_j$$

Le vecteur directeur de la droite recherché est alors vecteur propre associé à la plus grande valeur propre de la matrice de covariance des points

$$M = \sum_i (P_i . P_i^T)$$

L'interêt de cette formulation réside dans le fait qu'elle est indépendante de la dimension de l'image; elle s'applique également aux images 3-D.

Dans le cas 2-D on trouve la valeur propre

$$t1 = \frac{1}{2}(X+Y) + C \quad \text{où} \quad C = \sqrt{\frac{1}{2}(X-Y)^2 + Z^2}$$

$$\text{avec } X = \sum x_i^2 \quad Y = \sum y_i^2 \quad Z = \sum x_i y_i$$

Un vecteur propre associé est

$$x1 = Z/(t1-X) \quad y1 = 1$$

$$\text{ou } x1 = 1 \quad y1 = Z/(t1-Y)$$

Pour l'interprétation de ces résultats il est important de noter que la donnée d'une ligne ne nous permet pas de déduire une droite unique. De ce point de vue, l'approche syntaxique est plus correcte: un échantillon fini d'une droite digitale nous permet seulement de définir un "interval de segments de droites". Aucun segment dans cet interval n'est "meilleur" que les autres.

V Autres méthodes

V.1 Analyse de courbure

Si on dispose de la courbure de la ligne à découper, la détection des segments de droites est simple, puisqu'il suffit de trouver les parties à courbure nulle. Le problème majeur de cette approche vient du fait que le calcul de la courbure en tant que dérivée seconde est fortement affecté par le bruit.
Cette approche a effectivement été utilisée par Perkins [9], qui calcule la courbure à partir des résultats fournis par l'opérateur de Hueckel employé pour la construction des lignes.

V.2 Méthode de Hough

Une approche totalement différente est donné par les méthodes de Hough [2]. Elles permettent de résoudre un problème différent du notre: La détection de segment de droites dans un nuage quelconque de points, un segment pouvant être interrompu, ou présent en pointillé. La mise en oeuvre de ces méthodes est relativement onéreuse et nous ne l'approfondissons pas ici.

VI Découpage itératif

Les deux méthodes que nous avons présentées de façon détaillée - le découpage récursif et la reconnaissance syntaxique - ont des qualités et des défauts complémentaires. En dehors de quelques cas exceptionnels, le découpage récursif est peu sensible au bruit (à condition de bien choisir ses paramètres). Son inconvénient majeur vient d'un besoin en place mémoire et en temps d'exécution non-linéaire. La reconnaissance syntaxique a un caractère incrémental, donc une bonne efficacité. Par contre, elle accepte mal le bruit dû au capteur.

Nous avons donc cherché à combiner les avantages des deux méthodes et proposons l'algorithme de découpage itératif suivant:

1/ Test de linéarité.
Sur les EPSL premiers points de la ligne, on calcule la distance à la droite passant par le premier et le dernier point, comme pour le découpage récursif. Si Dx > EPSD on itère sur ce test avec la ligne (Pmax ... Pn). A la sortie de cette itération on dispose d'un segment (Pi ... Pi+EPSL-1) définissant une droite D.

2/ Vérification.
Si les points suivants - Pi+EPSL ... se trouvent à une distance inférieure à EPSD de la droite D, ils font partie du segment.

La structure de cet algorithme - détermination des paramètres, puis vérification - est celle de la reconnaissance syntaxique, la première étape étant empruntée au découpage récursif. Il est incrémental, ce qui imlique en particulier que

* la longueur des droites pouvant être détectées n'est pas bornée
* les résultats sont fournis au fur et à mesure de l'analyse de la ligne.

Il est possible de combiner cet algorithme avec une approximation aux moindres carrés. En effet, contrairement à l'apparence, toutes les valeurs intervenant dans les formules du paragraphe IV peuvent être calculées de façon incrémentale. Dans ce cas, les équations des résultats sont relativement indépendantes du choix des points initiaux.

VII Conclusion

Nous avons montré un algorithme de découpage d'une ligne digitale en segments de droites qui réunit plusieurs avantages importants:

* Il a un coût proportionnel au nombre de points à traiter.
* Il fournit les résultats au fur et à mesure.
* Les résultats satisfont un critère d'approximation aux moindres carrés.

Cet algorithme est utilisé depuis 1982 dans le système CAIMAN, et sert en particulier pour PVV [11], un système d'identification et de localisation d'objets. Nous comptons compléter notre étude par une mesure expérimentale de la précision des résultats par rapport à un repère externe.

Reférences

[1] Ayache. N.(1983)
Un système de vision bidimensionnelle en robotique industrielle
Université de Paris Sud

[2] Ballard, D.H., Brown, C.M.(1982)
Computer Vision
Prentice Hall, Englewood Cliffs, N.J.

[3] Brooks, R.A.(1981)
Symbolic reasoning among 3-D objects and 2-D models
Artificial Intelligence 17, p. 285-348

[4] Duda, R.O., Hart, P.E.(1973)
Pattern classification and scene analysis
John Wiley and Sons

[5] Fu, Lu (1977)
A clustering procedure for syntactic patterns
IEEE Trans., SMC-7, p.734-742

[6] Horaud, P.(1981)
Extraction et segmentation de contours dans une image - Applications à l'inspection automatique
Thèse de docteur ingénieur, ENSIEG, Grenoble

[7] Horn, B.K.P.(1972)
VISMEM: A bag of "robotics" formulae
Vision Flash 34, MIT

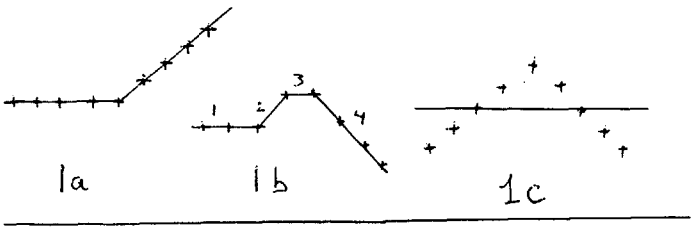
[8] Nevatia, R., Babu, K.R.(1979)
Linear feature extraction and description
IJCAI, p. 639-641

[9] Perkins, W.A.(1977)
Model based vision system for scenes containg multiple parts
IJCAI-77, p.678-684

[10] Rosenfeld, A.(1974)
Digital straight line segments
IEEE Transactions on Computers, vol.C-23, p.1264-1269

[11] Souvignier, V. (1983)
PVV - Un système d'interprétation d'images par prédiction - vérification
Thèse de troisième cycle, INP Grenoble

[12] Berlioux, P., Bizard, Ph.(1983)
Construction, preuve et évaluation de programmes
Dunod



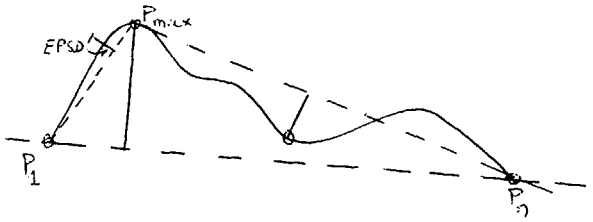


Fig.2 Découpage récursif - premiers points de cassure

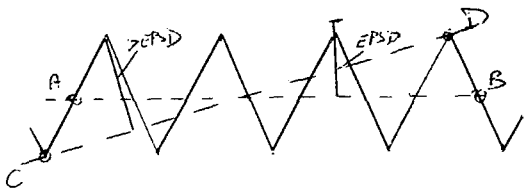


Fig.3 Instabilité du découpage récursif. Avec le choix de EPSD et EPSL donné, on obtient un segment à partir des points initiaux A,B, et 7 segments avec les points C et D.

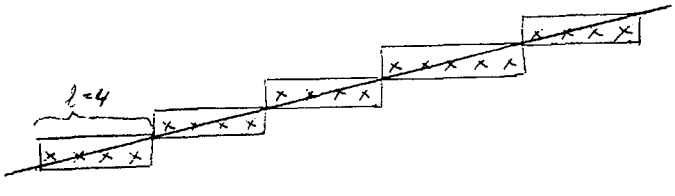


Fig.4 Les barrettes d'une droite digitale.
Codes direction: 00010001000100001000