Comment effectuer les cinq opérations de complétion classiques de la tomographie discrète en temps linéaire

Alain Daurat

Comment effectuer les cinq opérations de complétion classiques de la tomographie discrète en temps linéaire

Alain Daurat*

13 décembre 2006

Résumé

Les cinq opérations de complétion étudiées dans ce papier sont utilisées pour la reconstruction de convexes discrets à partir de leurs projections tomographiques. L'article [3] présente un algorithme effectuant ces cinq opérations avec une complexité $O(N^2 \log N)$ où N est la taille des projections. Ce papier présente une version modifiée de cet algorithme qui a une complexité en $O(N^2)$, soit un temps linéaire par rapport à la taille de l'image reconstruite. Ce résultat permet de montrer que les problèmes de reconstruction de convexes déjà connus comme polynomiaux ont en fait une complexité en $O(N^4)$.

1 Préliminaires

Étant donnée une partie finie E de $\mathbb{Z}^2$ la projection tomographique de E selon la direction p dirigée par le vecteur (a, b), avec a et b premiers entre eux, est la fonction $X_p : \mathbb{Z} \to \mathbb{N}$ définie par :

$$X_p E(k) = |\{M \in E : p(M) = k\}| \text{ pour tout } k \in \mathbb{Z}.$$

On s'intéresse à la reconstruction d'un ensemble à partir de ses projections tomographiques selon un nombre fini de directions. Dans cet article, on impose en plus que les ensembles reconstruits soient *convexes* selon les directions de projection : un ensemble E est dit convexe selon une direction, si son intersection avec toute droite parallèle à cette direction est constituée de points consécutifs. En effet il est facile de trouver un ensemble fini R qui contient tous les ensembles ayant les projections prescrites en regardant les supports des projections. On considère alors deux ensembles variables $\alpha \subset \beta \subset R$. Chaque point M peut alors être dans un des trois états suivant :

– $M \in \alpha$: le point est dans l'ensemble à reconstruire.
– $M \in R \setminus \beta$: le point n'est pas dans l'ensemble à reconstruire.

*LSIIT CNRS UMR 7005, Université Louis Pasteur (Strasbourg 1), Pôle API, Boulevard Sébastien Brant, 67400 Illkirch-Graffenstaden, France, daurat@dpt-info.u-strasbg.fr

– $M \in \beta \setminus \alpha$: on ne sait pas (encore) si le point M est dans l'ensemble à reconstruire.
Les opérations de complétion sont des opérations qui font augmenter α et diminuer β.
On considère les cinq opérations $\oplus$, $\ominus$, $\otimes$, $\odot$, $\odot'$. Pour leur définition précise voir [3]. Dans [3] est aussi décrit un algorithme de complexité $O(N^2 \log N)$. La suite présente un algorithme de complexité $O(N^2)$.

2 Un algorithme pour les opérations de complétion

Dans cette section, on suppose que l'ensemble de directions est seulement composée de la direction horizontale (h) et de la direction vertical v. La droite d'équation $x = j$ (resp. $y = i$) sera notée "$v = j$" (resp. "$h = i$"). De plus on utilise les conventions $\min \emptyset = +\infty$, $\max \emptyset = -\infty$. On se fixe donc deux vecteurs-projection $(h_i)_{1 \leq i \leq m}$ et $(v_j)_{1 \leq j \leq n}$, et on note $N = \max(m, n)$.

2.1 Les différentes structures utilisées

L'algorithme utilise un simple tableau de booléens pour implanter les ensembles α et β. Pour chaque droite horizontale $h = i$ on a les données suivantes :
– les 8 variables $l_1(\alpha_i^h)$, $l_2(\alpha_i^h)$, $r_1(\alpha_i^h)$, $r_2(\alpha_i^h)$, $l_1(\beta_i^h)$, $l_2(\beta_i^h)$, $r_1(\beta_i^h)$, $r_2(\beta_i^h)$ de sorte que tout au long de l'algorithme on a toujours les propriétés suivantes :
  – Tous les points (i, j) avec $j \in [l_1(\alpha), r_1(\alpha)]$ sont dans α.
  – $l_2(\alpha_i^h) = \min(\{j \ : \ (i, j) \in \alpha\})$, $r_2(\alpha_i^h) = \max(\{j \ : \ (i, j) \in \alpha\})$.
  – $l_1(\beta_i^h) = \min(\{j \ : \ (i, j) \in \beta\})$, $r_2(\beta_i^h) = \max(\{j \ : \ (i, j) \in \beta\})$.
  – $l_2(\beta_i^h) = \max(\{j \ : \ (i, j) \notin \beta_i^p \text{ and } j < l_2(\alpha_i^h)\}) + 1$, $r_2(\beta_i^h) = \min(\{j \ : \ (i, j) \notin \beta_i^p \text{ and } j > r_2(\alpha_i^h)\}) - 1$
– les variables $\texttt{card}\alpha_i^h$ et $\texttt{card}\beta_i^h$ qui sont toujours égales aux cardinaux des ensembles $\alpha_i^h = \{h = i\} \cap \alpha$ et $\beta_i^h = \{h = i\} \cap \beta$
– le tableau entier $\texttt{suiv\_dans\_beta}_i^h$, défini par : $\texttt{suiv\_dans\_beta}_i^h[j] = \min(\{k > j : (i, j) \in \beta_i^h\})$.
– le tableau entier $\texttt{prec\_dans\_beta}_i^h$ défini par : $\texttt{prec\_dans\_beta}_i^h[j] = \max(\{k < j : (i, j) \in \beta_i^h\})$.

On a aussi des données similaires pour les droites verticales. De plus on doit disposer de la structure `droites_a_traiter` qui mémorise les droites qui doivent être passées par les opérations de complétion. On utilise deux opérations sur cette structure :
– `estvide(droites_a_traiter)` qui indique si reste encore des lignes à traiter.
– `extrait(droites_a_traiter)` qui renvoie une des lignes à traiter, et l'enlève de l'ensemble

Sachant qu'il y a au maximum $m + n$ lignes à traiter, ces deux opérations peuvent être effectuées en temps constant. (voir [1] pour plus de détails).

2.2 Remises à jour

Les deux procédures suivantes indiquent comment sont remises à jour les structures décrites précédemment quand on rajoute un point à α ou lorsque l'on enlève un point à β. On distingue le cas ou la modification du point provient d'une opération de complétion sur une ligne horizontale ou sur une ligne verticale. (car l'ensemble `droites_a_traiter` est mis à jour différemment).

`met_dans_alpha`$_h(i,j)$

si $(i,j) \notin \beta$ **alors**
  ARRETER(aucune solution)
fin si
si $(i,j) \in \alpha$ **alors**
  retourne
fin si
$\alpha \leftarrow \alpha \cup \{(i,j)\}$
pour $(p,i',j') \in \{(h,i,j),(v,j,i)\}$ **faire**
  si `card`$\alpha^p_{i'} = 0$ **alors**
    // Premier point de α, les deux instructions suivantes prennent
    // temps $O(N)$ mais sont exécutées $O(N)$ fois (cette étape a été oubliée dans [3])
    $l_2(\beta^p_{i'}) \leftarrow \max(\{j'' : (i',j'') \notin \beta^p_{i'} \text{ and } j'' < j'\}) + 1$
    $r_2(\beta^p_{i'}) \leftarrow \min(\{j'' : (i',j'') \notin \beta^p_{i'} \text{ and } j'' > j'\}) - 1$
  fin si
  $l_2(\alpha^p_{i'}) \leftarrow \min(l_2(\alpha^p_{i'}), j')$
  $r_2(\alpha^p_{i'}) \leftarrow \max(r_2(\alpha^p_{i'}), j')$
  `card`$\alpha^p_{i'} \leftarrow$ `card`$\alpha^p_{i'} + 1$
fin pour
`ajouter_droite`(`droites_a_traiter`, $v = j$)

`enleve_de_beta`$_h(i,j)$

si $(i,j) \in \alpha$ **alors**
  ARRETER(aucune solution)
fin si
si $(i,j) \notin \beta$ **alors**
  retourne
fin si
$\beta \leftarrow \beta \setminus \{(i,j)\}$
pour $(p,i',j',x) \in \{(h,i,j,h_i),(v,j,i,v_j)\}$ **faire**
  si $j' = l_1(\beta^p_{i'})$ **alors**
    $l_1(\beta^p_{i'}) \leftarrow$ `suiv_dans_beta`$^p_{i'}[j']$
  fin si
  si $j' = r_1(\beta^p_{i'})$ **alors**
    $r_1(\beta^p_{i'}) \leftarrow$ `prec_dans_beta`$^p_{i'}[j']$
  fin si
  si `card`$\alpha^p_{i'} \neq 0$ **alors**
    si $j' < l_2(\alpha^p_{i'})$ **alors**
      $l_2(\beta^p_{i'}) \leftarrow \max(l_2(\beta^p_{i'}), j' + 1)$

fin si
si $j' > r_2(\alpha^p_{i'})$ **alors**
$r_2(\beta^p_{i'}) \leftarrow \min(l_2(\beta^p_{i'}), j' - 1)$
fin si
fin si
$\texttt{card}\beta^p_{i'} \leftarrow \texttt{card}\beta^p_{i'} - 1$
$\texttt{suiv\_dans\_beta}^p_{i'}[\texttt{prec\_dans\_beta}^p_{i'}[j']] \leftarrow \texttt{suiv\_dans\_beta}^p_{i'}[j']$
$\texttt{prec\_dans\_beta}^p_{i'}[\texttt{suiv\_dans\_beta}^p_{i'}[j']] \leftarrow \texttt{prec\_dans\_beta}^p_{i'}[j']$
fin pour
$\texttt{ajouter\_droite}(\texttt{droites\_a\_traiter}, v = j)$

Les procédures $\texttt{met\_dans\_alpha}_v(i,j)$, $\texttt{enleve\_de\_beta}_v(i,j)$ sont similaires.

2.3 Opérations de complétions sur une droite

La procédure suivante effectue les 4 premières opérations de complétion sur une droite horizontale :

$\texttt{traite\_droite1}(h = i)$

si $\texttt{card}\alpha^p_{i'} \neq 0$ **alors**
si $l_1(\alpha^h_i) \neq +\infty$ **alors** *// Opération* $\oplus$
pour tout $j \in [l_2(\alpha^h_i) + 1, r_2(\alpha^h_i) - 1]$ **faire**
$\texttt{met\_dans\_alpha}_h(i,j)$
fin pour
sinon
pour tout $j \in [l_2(\alpha^h_i) + 1, l_1(\alpha^h_i) - 1] \cup [r_1(\alpha^h_i) + 1, r_2(\alpha^h_i) - 1]$ **faire**
$\texttt{met\_dans\_alpha}_h(i,j)$
fin pour
fin si
$l_1(\alpha^h_i) \leftarrow l_2(\alpha^h_i)\,;\, r_1(\alpha^h_i) \leftarrow r_2(\alpha^h_i)$
pour tout $j \in [l_1(\beta^h_i), l_2(\beta^h_i) - 1] \cup [r_2(\beta^h_i) + 1, r_1(\beta^h_i)]$ **faire** *// Opération* $\ominus$
$\texttt{enleve\_de\_beta}_h(i,j)$
fin pour
fin si
si $r_1(\beta^h_i) - h_i + 1 \leq l_1(\beta^h_i) + h_i - 1$ **alors** *// Opération* $\otimes$
si $\texttt{card}\alpha^p_{i'} = 0$ **alors**
pour tout $j \in [r_1(\beta^h_i) - h_i + 1, l_1(\beta^h_i) + h_i - 1]$ **faire**
$\texttt{met\_dans\_alpha}_h(i,j)$
fin pour
sinon
pour tout $j \in [r_1(\beta^h_i) - h_i + 1, l_1(\alpha^h_i) - 1] \cup [r_1(\alpha^h_i) + 1, l_1(\beta^h_i) + h_i - 1]$ **faire**
$\texttt{met\_dans\_alpha}_h(i,j)$
fin pour
fin si
$l_1(\alpha^h_i) \leftarrow l_2(\alpha^h_i)\,;\, r_1(\alpha^h_i) \leftarrow r_2(\alpha^h_i)$
fin si
si $\texttt{card}\alpha^p_{i'} \neq 0$ **alors** *// Opération* $\odot$

pour tout $j \in [l_1(\beta_i^h), r_1(\alpha_i^h) - h_i] \cup [l_1(\alpha_i^h) + h_i, r_1(\beta_i^h)]$ **faire**
`enleve_de_beta`$_h(i,j)$
fin pour
fin si

Cette procédure effectue les 5 opérations sur une droite horizontale :
`traite_droite`$(h = i)$

`traite_droite1`$(h = i)$
tant que `card`$\beta_i^h < 2h_i -$ `card`α_i^h **faire**
$j_1 \leftarrow l_1(\beta_i^h)$; $j_2 \leftarrow r_1(\beta_i^h)$
// Le nombre d'itérations de cette boucle est égal au nombre de points enlevés
tant que $(i, j_1) \in \beta$ et $(i, j_2) \in \beta$ **faire**
$j_1 \leftarrow j_1 + 1$; $j_2 \leftarrow j_2 - 1$
fin tant que
si $(i, j_1) \notin \beta$ **alors**
pour tout $j \in [l_1(\beta_i^h), j_1 - 1]$ **faire**
`enleve_de_beta`$_h(i,j)$
fin pour
sinon
pour tout $j \in [j_2 + 1, r_1(\beta_i^h)]$ **faire**
`enleve_de_beta`$_h(i,j)$
fin pour
fin si
fin tant que
`traite_droite1`$(h = i)$

Les procédures `traite_droite1`$(v = j)$, `traite_droite`$(v = j)$ sont similaires.

2.4 Algorithme global

Voici l'algorithme exécutant les 5 opérations de complétion.
`operations_de_completion`(α_0, β_0)

$\alpha \leftarrow \alpha_0$; $\beta \leftarrow \beta_0$
$\beta \leftarrow \beta \setminus \{(i,j) \ : \ h_i = 0 \text{ or } v_j = 0\}$
pour tout $l \in \{h = i \ : \ 1 \le i \le m \text{ and } h_i > 0\} \cup \{v = j \ : \ 1 \le j \le n \text{ and } v_j > 0\}$ **faire**
`ajouter_droite`(`droites_a_traiter`$, l)$
fin pour
initialiser $l_1, l_2, r_1, r_2,$ `suiv_dans_beta`, `prec_dans_beta`, `card`α, `card`β pour toutes les droites de `droites_a_traiter`
tant que non(`estvide`(`droites_a_traiter`)) **faire**
$l \leftarrow$ `extrait`(`droites_a_traiter`)
`traite_droite`(l)
fin tant que
retourne(α, β)

2.5 Correction et complexité de l'algorithme

La différence essentielle avec l'algorithme de [3] est que l'opération $\odot'$ n'est effectuée sur les droites qui vérifient $|\beta_i^h| < 2h_i - |\alpha_i^h|$. Mais comme cette opération a été introduite pour que cette condition ne soit pas vérifiée sur chacune des lignes (voir par exemple [1, proposition 4.1]), il est suffisant d'appliquer cette opération seulement dans ces cas-là. Or lorsque $|\beta_i^h| < 2h_i - |\alpha_i^h|$ et que l'opération est stable par les 4 premières opérations de complétion alors on a forcément $|\alpha_i^h| = 0$ et β_i^h est composé d'au moins deux composantes connexes. Et au plus une de ces composantes a plus de h_i points. Donc l'opération $\odot'$ s'applique sur la composante la plus à gauche ou sur la composante la plus à droite. Ceci montre la correction de la procédure `traite_droite`. Le reste de l'analyse de l'algorithme peut se faire exactement de la même manière que dans [2]. De même en reprenant les mêmes arguments que dans [2] on peut voir l'algorithme décrit dans cette section est de complexité $O(N^2)$.

3 Conséquences

Si $\mathcal{M}$ est une classe de parties de $\mathbb{Z}^2$ et $\mathcal{D}$ est un ensemble fini de directions, alors on définit le problème algorithmique suivant : RECONSTRUCTION($\mathcal{M}, \mathcal{D}$)
Entrée : Une fonction $f : \mathcal{D} \times \mathbb{Z} \to \mathbb{N}$ à support fini
Sortie : Un ensemble fini $E \in MM$ tel que $X_pE(k) = f(p,k)$ pour tout $(p,k) \in \mathcal{D} \times \mathbb{Z}$
On peut voir que l'algorithme décrit à la section précédente se généralise à un ensemble quelconques de directions avec toujours une complexité de $O(N^2)$ où N est la tailles maximale des projections, on en déduit, en utilisant la méthode de reconstruction décrite dans [4] :

Théorème 1 *Si $\mathcal{D} = \{h, v\}$ et $\mathcal{P}$ est la classe des polyominos HV-convexes alors* RECONSTRUCTION($\mathcal{P}, \mathcal{D}$) *peut être résolu en $O(N^4)$ où N est le maximum des tailles des projections horizontales et verticales.*

Ce résultat était déjà connu, mais avec algorithme qui n'utilisait pas d'opérations de complétion. [5]
En utilisant la méthode décrite dans [2], on peut aussi montrer :

Théorème 2 *Si $\mathcal{D}$ est un ensemble de directions dont 4 d'entre-elles ont un birapport qui n'est pas dans l'ensemble $\{4/3, 3/2, 2, 3, 4\}$ et $\mathcal{C}$ est la classes des convexes discrets (intersections de polygones convexes avec $\mathbb{Z}^2$ alors* RECONSTRUCTION($\mathcal{C}, \mathcal{D}$) *peut être résolu en temps $O(N^4)$, où $N = \max_{p\in\mathcal{D}}(\max(\{k : f(p,k) > 0\}) - \min(\{k : f(p,k) > 0\}) + 1)$.*

Références

[1] S. Brunetti et A. Daurat. Reconstruction of discrete sets from two or more X-rays in any direction. Dans *Proc. of IWCIA 2000*, pages 241–258. Université de Caen, 2000.

[2] S. Brunetti et A. Daurat. Reconstruction of Q-convex sets. Dans G. T. Herman et A. Kuba, editors, *Advances in Discrete Tomography and its Applications*. Birkhäuser, à paraître.

[3] S. Brunetti, A. Daurat, et A. Kuba. Fast filling operations used in the reconstruction of convex lattice sets. Dans *Proc. of DGCI 2006*, volume 4245 de *Lecture Notes in Comp. Sci.*, pages 98–109, 2006.

[4] S. Brunetti, A. Del Lungo, F. Del Ristoro, A. Kuba, et M. Nivat. Reconstruction of 4- and 8-connected convex discrete sets from row and column projections. *Linear Algebra Appl.*, 339 :37–57, 2001.

[5] M. Chrobak et C. Dürr. Reconstructing hv-convex polyominoes from orthogonal projections. *Inform. Process. Lett.*, 69 :283–289, 1999.

[6] M. Gębala. The reconstruction of convex polyominoes from horizontal and vertical projections. Dans *Proc. of SOFSEM '98*, volume 1521 de *Lecture Notes in Comp. Sci.*, pages 350–359, 1998.