Nouvelle Approche Efficace pour la Reconnaissance de Plans Discrets

4.1 Introduction

Comme nous l'avons vu dans le chapitre 3, la reconnaissance de plans discrets est un problème largement étudié en géométrie discrète. Nous décrivons dans ce chapitre une nouvelle méthode pour décider si un ensemble de n points de $\mathbb{Z}^3$ correspond à la discrétisation d'un morceau de plan euclidien. La méthode que nous décrivons ici est celle proposée à la conférence internationale DGCI (Discrete Geometry for Computer Imagery) de 2008 (voir [CB08a]). Une version préliminaire de cet algorithme avait déjà été présentée en 2006 par Lilian Buzer dans [Buz06].

La plupart des méthodes existantes ne combinent pas à la fois une complexité intéressante dans le pire cas et une efficacité en pratique. Par exemple, les méthodes utilisant les résultats optimaux de Megiddo (voir [Meg84]) ont beau être linéaires, elles peuvent difficilement être utilisées en pratique. Quant à la méthode des cordes (voir [GDRZ05]), elle parvient à traiter un ensemble de 10^6 points en parcourant environ 10 fois l'ensemble des cordes mais sa complexité dans le pire cas est de l'ordre de $O(n^7)$. La méthode de 2006 de Lilian Buzer [Buz06] atteint une complexité dans le pire cas de $O(n \log^2 D)$ où $D-1$ représente la plus grande longueur d'une boite englobant l'ensemble de points et elle reconnaît un ensemble dense de 10^6 points en environ 360 itérations linéaires. Notre méthode actuelle est plus efficace en pratique puisqu'elle reconnaît un ensemble dense de 10^6 points en environ 10 itérations linéaires et elle atteint une complexité dans le pire cas de $O(n \log D)$.

La méthode de 2006 et la méthode présentée ici transforment le problème de reconnaissance de plans discrets naïfs en un problème de réalisabilité sur une fonction convexe bi-dimensionelle dite *fonction épaisseur*. Par conséquent, ces méthodes ne prennent en compte que deux paramètres et nous pouvons utiliser des techniques de géométrie discrète planaire afin de déterminer si l'espace des solutions 2D est vide ou pas. Pour résoudre le problème de réalisabilité, notre approche actuelle utilise une propriété du centre de gravité tandis que la méthode de 2006 combine l'oracle de Megiddo et une recherche dichotomique mono-dimensionnelle. Ce principal changement améliore la complexité en temps de la méthode et diminue considérablement son nombre d'itérations. Notre algorithme peut également être conçu de façon incrémentale.

Dans la section 4.2, nous montrons comment le problème de reconnaissance de plans discrets peut être transformé en un problème de réalisabilité sur une fonction convexe bi-

dimensionnelle, la fonction épaisseur. Nous introduisons dans la section 4.3 une méthode pour résoudre ce problème de réalisabilité en utilisant le calcul d'un sous-gradient de la fonction épaisseur. Pour résoudre le problème de réalisabilité, nous réduisons l'espace de recherche continu en appliquant itérativement des coupes jusqu'à trouver un point solution ou jusqu'à ce que l'espace de recherche devienne vide. Dans la section 4.4, nous décrivons cette étape de réduction et nous commentons sont efficacité. Pour améliorer notre algorithme, nous discrétisons le domaine de recherche du problème de réalisabilité. Nous justifions dans la section 4.5 le choix du pas de discrétisation et nous expliquons comment adapter notre méthode à un domaine de recherche discrétisé. Nous analysons la complexité en temps de notre méthode dans la section 4.6. Dans la section 4.7, nous décrivons la version incrémentale de notre algorithme et nous terminons ce chapitre avec quelques résultats expérimentaux et la description d'améliorations pratiques.

4.2 La Reconnaissance de Plans Comme un Problème de Réalisabilité

Dans la section 3.2, nous avons défini la fonction épaisseur par rapport à un ensemble de points de $\mathbb{Z}^3$ noté S comme une fonction de $[-1,1]^2$ vers $\mathbb{R}^+$ définie par :

$$ep_S(u) = \max_{1\leq i\leq n}(N_u \cdot p^i) - \min_{1\leq i\leq n}(N_u \cdot p^i) \tag{4.1}$$

où $u = (u_1, u_2)$ appartient à $[-1,1]^2$ et N_u correspond au vecteur $(u_1, u_2, 1)$.

D'après la proposition 4, l'ensemble de points S correspond à un morceau de plan discret s'il existe un vecteur u de $[-1,1]^2$ qui vérifie $ep_S(u) < 1$. Notre problème de reconnaissance de plans discrets revient donc à déterminer un tel vecteur u ou à montrer qu'aucun vecteur de $[-1,1]^2$ ne satisfait $ep_S(u) < 1$. Par conséquent, nous nous ramènons à un problème de réalisabilité sur la fonction bi-dimensionnelle ep_S.

Proposition 6 *La fonction ep_S est une fonction convexe.*

Preuve 7 *Soit N_u le vecteur associé au vecteur $u \in [-1,1]^2$. Considérons la fonction $g^i(u)$ de $[-1,1]^2$ vers $\mathbb{R}$ qui associe au vecteur u la valeur $N_u \cdot p^i$. Cette fonction est une fonction affine, elle est donc convexe. Remarquons à présent que la fonction $ep_S(u)$ peut être ré-écrite comme $\max_{1\leq i\leq n}(g^i(u)) + \max_{1\leq i\leq n}(-g^i(u))$. Comme le maximum des fonctions $(g^i)_{1\leq i\leq n}$ est également convexe et comme la somme de deux fonctions convexes est convexe, nous en déduisons que la fonction ep_S est elle-même convexe.*

Le problème de reconnaissance de plans discrets est équivalent à un problème de réalisabilité sur une fonction convexe bi-dimensionnelle de domaine de recherche $[-1,1]^2$. Si l'ensemble S ne correspond pas à un morceau de plan discret naïf alors l'espace des solutions du problème de réalisabilité est vide. Sinon, l'espace des solutions correspond à un polygone convexe non-vide inclus dans $[-1,1]^2$.

4.3 Résoudre le Problème de Réalisabilité

4.3.1 Introduction Générale

Remarquons tout d'abord que nous ne traitons pas un problème d'optimisation mais un problème de réalisabilité. Ceci signifie que nous ne souhaitons pas déterminer d'un point u

dans le domaine de recherche tel que la valeur $ep_S(u)$ est minimum mais nous recherchons un point u tel que $ep_S(u)$ est strictement inférieure à 1. Cependant, certaines méthodes utilisées en optimisation convexe peuvent être adaptées pour résoudre des problèmes de réalisabilité.

De façon générale, pour optimiser une fonction convexe $f : \mathbb{R}^d \to \mathbb{R}$, nous devons déterminer le point x pour lequel la valeur $f(x)$ est minimum ou maximum. Pour cela, nous pouvons utiliser la méthode de descente du gradient. En utilisant cette méthode, nous approchons le minimum de la fonction en nous déplaçant itérativement dans la direction de plus forte pente. À l'itération i, le gradient $\nabla(x^i)$ de f au point x^i est calculé. Par définition du gradient, il existe une valeur réelle τ_i telle que $f(x^i)$ est inférieure à $f(x^i + \tau_i \nabla(x^i))$. Le point x^{i+1} de référence à l'itération suivante est alors égal à :

$$x^{i+1} = x^i + \tau_i \nabla(x^i) \tag{4.2}$$

Cependant, dans un souci d'efficacité, nous pouvons difficilement appliquer ce type d'algorithmes pour résoudre notre problème de réalisabilité. En effet, pour procéder à des calculs numériques exacts, les valeurs successives τ_i et par conséquent les coordonnées des points x^i doivent être représentées par des nombres rationnels. D'après (4.2), la taille du numérateur et du dénominateur de ces rationnels augmenteraient à chaque itération et ceci ralentirait significativement le calcul. Nous proposons donc une autre approche qui évite ce problème.

Notre approche pour résoudre le problème réalisabilité consiste à réduire itérativement le domaine de recherche jusqu'à ce que nous trouvions un point solution ou jusqu'à ce que le domaine de recherche soit vide. Cette réduction du domaine de recherche est faite grâce au calcul d'un sous-gradient de la fonction épaisseur.

4.3.2 Calcul du Sous-Gradient

Nous rappelons tout d'abord qu'un *sous-gradient* $g \in \mathbb{R}^d$ d'une fonction convexe $f : \mathbb{R}^d \to \mathbb{R}$ au point x vérifie : pour tout $x' \in \mathbb{R}^d$, $f(x') - f(x) \geq g \cdot (x' - x)$. Le sous-gradient est un vecteur qui indique la direction de plus forte pente de la fonction f au point x. Ceci signifie que si nous savons calculer un sous-gradient de la fonction ep_S en un point donné, nous connaissons un moyen de réduire le domaine de recherche. En effet, soit g_u un sous-gradient de la fonction ep_S au point u, alors pour tout vecteur $v \in [-1, 1]^2$ satisfaisant $g_u \cdot v \geq g_u \cdot u$, nous avons $ep_S(v) \geq ep_S(u)$. Si $ep_S(u)$ est plus grand que 1 alors $ep_S(v)$ est également plus grand que 1 et nous pouvons donc retirer tous ces points v du domaine de recherche. Les points à retirer correspondent à un des demi-plans portés par la droite de vecteur normal g_u passant par le point u. Vous remarquerez que nous utilisons le terme "sous-gradient" à la place du terme "gradient" car la fonction épaisseur n'est pas dérivable en tout point.

Pour résoudre notre problème de réalisabilité dans $\mathbb{R}^2$, nous devons déterminer comment calculer un sous-gradient de la fonction convexe ep_S à un point u donné. En accord avec la définition du sous-gradient, nous devons déterminer g tel que pour tout $v \in [-1, 1]^2$ nous avons :

$$ep_S(v) - ep_S(u) \geq g \cdot (v - u) \tag{4.3}$$

D'après la définition de ep_S, nous savons que $ep_S(u)$ est égal à $\max_{1 \leq i \leq n}(N_u \cdot p^i) - \min_{1 \leq i \leq n}(N_u \cdot p^i)$. Par conséquent, il existe deux points particuliers de S qui sont associés au produit scalaire maximum et au produit scalaire minimum. Notons respectivement p^a et p^b ces deux points (comme sur la figure 4.1), nous avons $ep_S(u) = N_u \cdot (p^a - p^b)$.

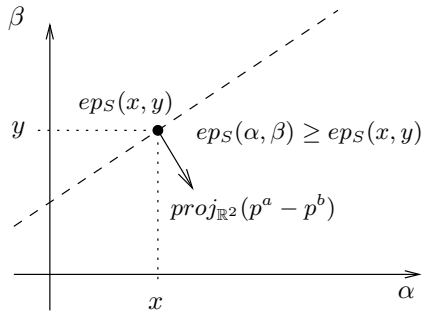


FIG. 4.1 – Un sous-gradient de ep_S

Soit T l'ensemble de deux points $\{p^a, p^b\}$. Comme T est inclus dans S, pour tout point $v \in [-1,1]^2$, l'épaisseur par rapport à l'ensemble T au point v est plus petite que l'épaisseur par rapport à l'ensemble S au point v. Nous avons :

$$\forall v \in [-1,1]^2, ep_T(v) \leq ep_S(v) \tag{4.4}$$

Comme l'ensemble T ne contient que deux points, la valeur $ep_T(v)$ est forcément égale à $|N_v \cdot (p^a - p^b)|$. De plus, nous remarquons que tout vecteur v peut être écrit comme $v + u - u$ et il s'en suit :

$$\begin{aligned} ep_T(v) &= |N_{v-u+u} \cdot (p^a - p^b)| \\ &= |N_u \cdot (p^a - p^b) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b)| \\ &= |ep_T(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b)| \end{aligned}$$

D'après (4.4) et par définition de la valeur absolue, nous obtenons :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_T(v) \geq ep_T(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b) \tag{4.5}$$

Nous rappelons que nous pouvons remplacer $ep_T(u)$ par $ep_S(u)$ et nous avons donc :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_S(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b) \tag{4.6}$$

D'après (4.3), nous pouvons à présent conclure que l'expression $proj_{\mathbb{R}^2}(p^a - p^b)$ est un sous-gradient de ep_S au point u (voir la figure 4.1). Cette valeur correspond à la projection du vecteur $p^a p^b$ dans $\mathbb{R}^2$. Nous constatons donc que les deux premières coordonnées du vecteur $p^a p^b$ sont suffisantes pour déterminer localement la variation de la fonction ep_S. Lorsque nous calculons $ep_S(u)$, nous déduisons indirectement les deux points p^a et p^b. Nous calculons donc en temps constant un sous-gradient.

Remarque 4 *Nous remarquons que si les deux points p^a et p^b associés respectivement à la valeur maximum et à la valeur minimum de la fonction épaisseur en un point ont les deux mêmes premières coordonnées, alors le sous-gradient $proj_{\mathbb{R}^2}(p^a - p^b)$ correspond au vecteur nul. Dans ce cas, le sous-gradient n'indique pas la direction de plus forte pente. Cependant, si l'ensemble de points S contient au moins deux points distincts avec les deux mêmes premières composantes, alors l'ensemble S ne peut pas correspondre à un morceau de plan discret. Par conséquent, si nous calculons un sous-gradient nul, nous pouvons en déduire que l'ensemble de points ne correspond pas à un morceau de plan discret.*

4.4 Réduction du Domaine de Recherche

Nous introduisons le principe général de notre méthode. Nous appelons *coupe valide* du domaine de recherche une coupe par un demi-plan qui préserve toutes les solutions réalisables. Le principe de notre méthode est d'appliquer itérativement des coupes valides sur le domaine de recherche $[-1, 1]^2$ afin de le réduire tant qu'aucune solution réalisable est trouvée. Si le domaine est réduit à l'ensemble vide, alors le problème n'a pas de solution.

4.4.1 Coupes par le Centre de Gravité

À chaque itération, nous calculons la valeur de la fonction ep_S en un point donné u. Si $ep_S(u)$ est strictement inférieure à 1, alors u correspond à une solution de notre problème de réalisabilité et le problème est donc résolu. Dans le cas contraire, nous appliquons une coupe sur le domaine de recherche. La droite de coupe passe par le point u et elle est orthogonale au sous-gradient de la fonction épaisseur au point u. La coupe est définie par l'inégalité suivante :

$$\nabla ep_S(u) \cdot v < \nabla ep_S(u) \cdot u \tag{4.7}$$

Par définition du sous-gradient, il s'en suit que tout point v éliminé par la coupe vérifie $ep_S(v) \geq 1$. Comme chacun de ces points ne correspond pas à une solution réalisable, nous en concluons que ces coupes sont valides.

À présent, nous souhaitons nous assurer de l'efficacité de chaque coupe. En effet, nous souhaitons que chaque coupe élimine au moins un pourcentage constant du domaine de recherche courant. Pour cela, nous choisissons d'évaluer la fonction épaisseur, et donc de générer la coupe valide, en un point particulier : le *centre de gravité* $C = (C_1, C_2)$ du domaine de recherche. Nous rappelons que le centre de gravité d'un polygone est équivalent à son *barycentre*. Nous calculons ses coordonnées en temps linéaire par rapport au nombre de sommets du polygone.

La proposition suivante assure l'efficacité de chaque coupe (voir [Neu45, Grü60]).

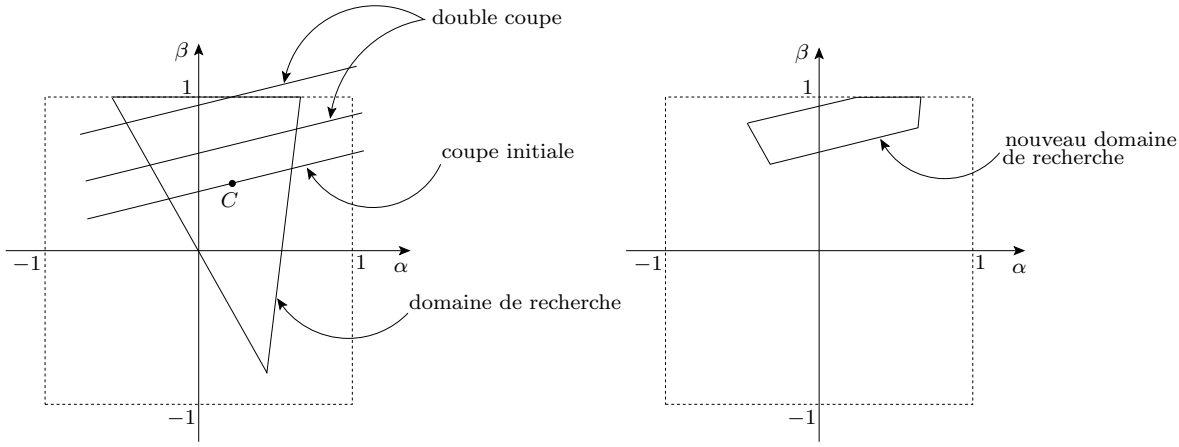
Proposition 8 *Soit K un polygone convexe dans le plan et soit C son centre de gravité. Chaque demi-plan dont la droite support passe par le centre de gravité C contient entre $4/9$ et $5/9$ de l'aire de K.*

Ceci signifie que chaque coupe passant par le centre de gravité du domaine de recherche élimine au moins $4/9$ de son aire. C'est donc au plus $5/9$ du domaine de recherche qui reste après la coupe. D'après [Neu45], nous savons que ce ratio est optimal dans le cas bi-dimensionnel. Nous pouvons remarquer qu'il est très proche du ratio $1/2$ de la dichotomie en dimension 1.

4.4.2 Amélioration des Coupes

Nous avons montré précédemment que la connaissance du sous-gradient nous permettait d'éliminer une partie du domaine de recherche. De plus, nous avons prouvé que chaque coupe passant par le centre de gravité du domaine de recherche courant éliminait au moins $4/9$ de son aire. Cependant, nous n'utilisons pas toute l'information disponible. En fait, lorsque nous calculons la valeur de $ep_S(C)$, nous déterminons deux points $p^a = (p_1^a, p_2^a, p_3^a)$ et $p^b = (p_1^b, p_2^b, p_3^b)$ de S tels que $ep_S(C) = |N_u \cdot (p^a - p^b)|$. Notons T l'ensemble de points $\{p^a, p^b\}$. Nous avons :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_T(v) \tag{4.8}$$



(a) La double coupe (b) Après la double coupe

FIG. 4.2 – Double coupe du domaine de recherche

Par conséquent, nous pouvons restreindre le domaine de recherche aux vecteurs $v = (v_1, v_2)$ qui vérifient $|N_v \cdot (p^a - p^b)| < 1$. Par définition de la valeur absolue, l'inégalité $|N_v \cdot (p^a - p^b)| < 1$ est équivalente à $N_v \cdot (p^a - p^b) < 1$ et $-N_v \cdot (p^a - p^b) < 1$. Si la valeur de $ep_S(C)$ est strictement inférieure à 1, le problème est résolu. Sinon, le domaine de recherche peut être réduit en appliquant une double coupe. Soit $\nabla ep_S(C) = proj_{\mathbb{R}^2}(p^a - p^b)$ le sous-gradient de ep_S au point C, les deux coupes sont définies par les deux inégalités suivantes (voir l'exemple en figure 4.3) :

$$\nabla ep_S(C) \cdot v < 1 - (p_3^a - p_3^b) \tag{4.9}$$

$$-\nabla ep_S(C) \cdot v < 1 + (p_3^a - p_3^b) \tag{4.10}$$

Le fait de générer ces deux coupes à chaque itération accélère bien sûr l'algorithme. Néanmoins, nous n'avons aucun moyen de quantifier la portion du domaine de recherche éliminée grâce à la double coupe par rapport à la coupe simple, la complexité de la méthode est donc inchangée.

4.5 Discrétisation du Domaine de Recherche

En appliquant une coupe simple ou une double coupe sur le domaine de recherche, nous nous assurons d'éliminer au moins 4/9 de l'aire du domaine de recherche courant à chaque itération. Néanmoins, cet argument n'est pas toujours suffisant pour assurer la terminaison de l'algorithme. En effet, si le problème de réalisabilité n'admet pas de solution, l'algorithme ne déterminera jamais un domaine de recherche vide car le domaine de recherche est continu. Nous discrétisons donc le domaine de recherche afin d'assurer la terminaison de l'algorithme et pouvoir évaluer la complexité.

4.5.1 Étude de l'Espace des Solutions

Soit F l'ensemble des solutions du problème de réalisabilité. L'ensemble F est un polygone convexe et il correspond aux points u de $[-1, 1]^2$ qui satisfont $ep_S(u) < 1$. Nous prouvons par la suite que si F n'est pas vide, alors il contient un carré de côté $1/(2D^3)$ où $D-1$ correspond à la plus grande longueur d'une boite englobant l'ensemble de points

S. Cette propriété nous permet de restreindre l'espace de recherche à une grille régulière de pas de discrétisation $1/(2D^3)$ sur $[-1,1]^2$.

Nous supposons que l'ensemble de points S est contenu dans une boite englobante de taille $D-1$ et que l'origine du repère est située au centre de cette boite englobante. De ce fait, tout point p^i de S vérifie $||p^i||_\infty \leq D/2$ pour $1 \leq i \leq n$ et tout vecteur k dont les extrémités correspondent à deux points de S vérifie :

$$||k||_\infty \leq D-1 \tag{4.11}$$

Considérons l'enveloppe convexe de l'ensemble S. D'après [DR95], nous savons que la fonction épaisseur atteint sa valeur minimum pour un vecteur particulier $\overline{N} = (\overline{N}_1, \overline{N}_2, \overline{N}_3)$ tel que chacun des deux plans supports de l'ensemble S de vecteur normal $\overline{N}$ passe par au moins deux sommets de l'enveloppe convexe. Par conséquent, le vecteur $\overline{N}$ correspond au produit vectoriel de deux vecteurs dont les extrémités correspondent à ces quatre sommets (voir [PS85]). D'après (4.11), nous avons $||\overline{N}||_\infty < 2D^2$ et en particulier :

$$0 \leq \overline{N}_3 < 2D^2 \tag{4.12}$$

D'après la définition 13, si S correspond à un morceau de plan discret naïf alors $\overline{N}$ vérifie $\mu \leq \overline{N}_1 p_1^i + \overline{N}_2 p_2^i + \overline{N}_3 p_3^i \leq \mu + \overline{N}_3 - 1$ pour tout $p^i = (p_1^i, p_2^i, p_3^i)$ de S. Soit $(\overline{N}'_1, \overline{N}'_2)$ le vecteur $(\overline{N}_1/\overline{N}_3, \overline{N}_2/\overline{N}_3)$, $(\overline{N}'_1, \overline{N}'_2)$ appartient à l'espace des solutions et nous avons :

$$\forall (p_1^i, p_2^i, p_3^i) \in S,\ \mu' \leq \overline{N}'_1 p_1^i + \overline{N}'_2 p_2^i + p_3^i \leq \mu' + 1 - 1/\overline{N}_3 \tag{4.13}$$

Nous pouvons remarquer qu'en faisant légèrement varier les deux premières coordonnées du vecteur $\overline{N}$, l'ensemble S correspond toujours à la discrétisation d'un morceau de plan euclidien de vecteur normal $\overline{N}$. Soit $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ le vecteur de la forme $(\overline{N}'_1 \pm \Delta, \overline{N}'_2 \pm \Delta)$ avec Δ une valeur réelle positive. Nous montrons que si $\Delta \leq 1/(4D^3)$ alors l'ensemble S correspond à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ et donc le vecteur $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ appartient lui aussi à l'espace des solutions. Pour cela, nous proposons de prouver la contraposée : si S ne correspond pas à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ alors $\Delta > 1/(4D^3)$.

D'après (4.11) et (4.13), les inégalités suivantes sont valides :

$$\mu' - D\Delta \leq \overline{N}_{\Delta 1} p_1^i + \overline{N}_{\Delta 2} p_2^i + p_3^i \leq \mu' + 1 - 1/N_3 + D\Delta \tag{4.14}$$

Ce qui est équivalent à :

$$\mu' - D\Delta \leq \overline{N}_{\Delta 1} p_1^i + \overline{N}_{\Delta 2} p_2^i + p_3^i \leq \mu' - D\Delta + 1 + 2D\Delta - 1/\overline{N}_3 \tag{4.15}$$

D'après la définition 13 et d'après (4.15), nous déduisons que si (4.15) ne définit pas un plan discret naïf alors l'expression $2D\Delta - 1/\overline{N}_3$ est positive et donc :

$$\Delta \geq 1/(2D\overline{N}_3) \tag{4.16}$$

Il s'en suit d'après (4.12) et (4.16) que dans ce cas :

$$\Delta > 1/(4D^3) \tag{4.17}$$

Nous en déduisons que si $\Delta \leq 1/(4D^3)$ alors l'ensemble S correspond à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ et donc le vecteur $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ appartient

lui aussi à l'espace des solutions. Par conséquent, nous pouvons restreindre le domaine de recherche à une grille régulière G de pas de discrétisation $1/(2D^3)$ sur $[-1,1]^2$, c'est-à-dire un carré de côté $4D^3$. En effet, pour tout ensemble de points S inclus dans une boite de longueur $D-1$ et correspondant à un morceau de plan discret naïf, il existe un carré Q de côté $1/(2D^3)$ dans $[-1,1]^2$ tel que tout point w de Q vérifie $ep_S(w) < 1$. Le choix du pas de discrétisation assure donc qu'au moins un point de la grille appartient à ce carré Q. Si aucun des points de l'échantillonnage ne correspond à une solution du problème de réalisabilité, alors l'espace des solutions est vide.

4.5.2 Amélioration du Domaine de Recherche Initial

Partir d'un domaine de recherche initial carré de côté $4D^3$ n'est pas toujours utile. Nous pouvons améliorer l'initialisation du domaine de recherche. Comme nous supposons que l'ensemble de points S est contenu dans une boite englobante de côté $D-1$, nous pouvons également supposer que certains points de S sont situés sur les bords de la boite englobante. Notons $p^l = (-D/2, p_2^l, p_3^l)$ et $p^r = (D/2, p_2^r, p_3^r)$ deux de ces points. Nous supposons que $p_2^l = p_2^r$. Ces deux points nous fournissent une contrainte supplémentaire sur l'espace des solutions. Si l'ensemble de points S correspond à un morceau de plan discret naïf de vecteur normal $N = (N_1, N_2, N_3)$, nous avons : $\gamma' \leq N \cdot (-D/2, p_2^l, p_3^l) < \gamma' + 1$ et $\gamma' \leq N \cdot (D/2, p_2^r, p_3^r) < \gamma' + 1$. Ceci implique que : $|N \cdot (D, 0, p_3^r - p_3^l)| < 1$. Nous pouvons donc en déduire la borne supérieure et inférieure suivante pour la coordonnée N_1 :

$$\frac{p_3^l - p_3^r}{D} - \frac{1}{D} < N_1 < \frac{p_3^l - p_3^r}{D} + \frac{1}{D} \tag{4.18}$$

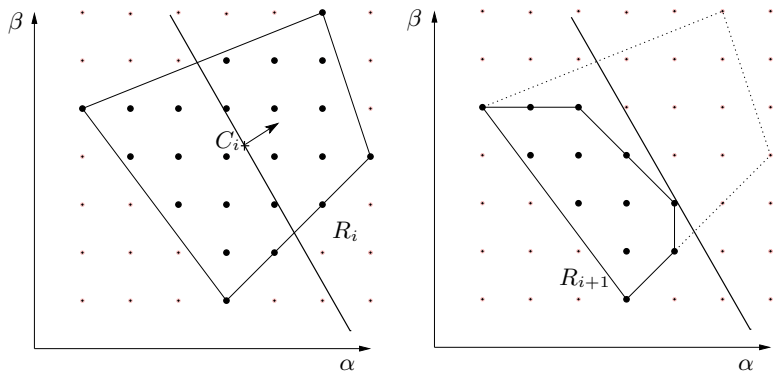
De la même manière, nous pouvons déterminer une borne supérieure et inférieure pour la coordonnée N_2 par rapport aux deux points $p^d = (p_1^d, -D/2, p_3^d)$ et $p^u = (p_1^u, D/2, p_3^u)$ de S tels que $p_1^d = p_1^u$:

$$\frac{p_3^d - p_3^u}{D} - \frac{1}{D} < N_2 < \frac{p_3^d - p_3^u}{D} + \frac{1}{D} \tag{4.19}$$

La taille des côtés du domaine de recherche est donc divisée par D. Cette étape nécessite simplement de parcourir l'ensemble de points S afin de déterminer les points extrêmes.

4.5.3 Domaine de Recherche Après Chaque Coupe

À chaque itération, nous appliquons une double coupe sur le domaine de recherche. Comme le domaine de recherche est initialisé comme un polygone convexe, il correspond toujours à un polygone convexe après chaque coupe. Néanmoins, les points d'intersection de la droite de coupe et des côtés du domaine de recherche courant correspondent à des points à coordonnées rationnelles. Par conséquent, le numérateur et le dénominateur de ces coordonnées rationnelles augmentent à chaque itération et cela a pour conséquence de ralentir l'exécution de l'algorithme. En outre, la terminaison de l'algorithme n'est pas assurée. Nous choisissons donc de décrire le domaine de recherche comme un polygone convexe à sommets portés par la grille, tout au long du déroulement de la méthode. Soit R_i le domaine de recherche au début de l'itération i. Il correspond à un polygone convexe dont les sommets sont portés par la grille. Notons à présent R'_i le domaine de recherche après la double coupe. Il correspond à un polygone convexe dont les sommets ont des coordonnées rationnelles. Soit $\overline{R'_i}$ le plus grand polygone convexe inclus dans R'_i à sommets portés par la grille du domaine de recherche. Par définition de $\overline{R'_i}$, aucun point



(a) Le domaine de recherche R_i d'aire 19.

(b) Le domaine de recherche réduit R_{i+1} d'aire 7.5 ($7.5 \leq (5/9) * 19$).

FIG. 4.3 – Coupe de R_i passant par le centre de gravité C_i

de la grille ne se situe dans $R'_i \backslash \overline{R'_i}$. Comme nous recherchons un point solution porté par la grille, nous remplaçons R_{i+1} par $\overline{R'_i}$ (voir la figure 4.3). Pour obtenir $\overline{R'_i}$ depuis R'_i, nous pouvons utiliser la méthode décrite dans la section 2.7.2 du chapitre 2 (voir aussi [CB09]) ou l'algorithme d'Harvey (voir [Har99]). Ces deux méthodes s'exécutent en $O(\log(D))$ dans le pire cas et sont efficaces en pratique.

4.6 Algorithme Résumé et Analyse de Complexité

Dans cette section, nous analysons la complexité en temps de notre algorithme de reconnaissance de plans discrets. Comme expliqué dans la section 4.4, que nous réduisions le domaine de recherche en appliquant une coupe simple ou une coupe double, cela n'a pas de conséquence sur la complexité de la méthode. Pour simplifier l'analyse de complexité, nous considérons donc que nous appliquons seulement une coupe simple sur le domaine de recherche à chaque itération, cette coupe passant par le centre de gravité du domaine de recherche courant.

4.6.1 Résumé de l'Algorithme

La toute première étape de l'algorithme consiste à initialiser le domaine de recherche R_1 comme un rectangle d'après les bornes décrites dans la section 4.5.2. Ensuite, tant que le domaine de recherche R_i contient au moins un point de la grille, l'algorithme calcule son centre de gravité C_i et il évalue la fonction épaisseur au point C_i par le biais d'un oracle. Si $ep_S(C_i)$ est strictement inférieure à 1, l'oracle retourne la valeur "vrai" et l'algorithme se termine. Ceci signifie que l'ensemble S correspond à un morceau de plan discret. Si $ep_S(C_i)$ est supérieure à 1, l'oracle calcule un sous-gradient ∇_i de la fonction épaisseur au point C_i et retourne la valeur "faux". Dans ce cas, une coupe est appliquée sur le domaine de recherche courant et l'algorithme réitère. Quand le domaine de recherche est vide, la boucle principale et donc l'algorithme se terminent. Ceci signifie que l'ensemble S ne correspond pas à un morceau de plan discret.

Durant le déroulement de l'algorithme, nous mettons à jour la liste des sommets du domaine de recherche courant R_i. Ainsi, nous pouvons connaître le nombre de ses sommets à tout moment en appelant la fonction *NombreDeSommets*. La fonction *Coupe*(R_i, C_i, ∇_i)

applique une coupe de vecteur normal ∇_i passant par le point C_i et il reconstruit le nouveau domaine de recherche R_{i+1} en utilisant la méthode de reconstruction décrite dans [CB09]. L'algorithme 15 résume notre méthode.

Algorithme 15 : Notre algorithme de reconnaissance de plans discrets

```
RECONNAISSANCE(S : ensemble de points 3D)
Valeur retournée : booléen
1  i ← 1   R_i ←InitDomaineDeRecherche(S)
2  TANT QUE NombreDeSommets(R_i) ≥ 1
3     C_i ←CentreDeGravité(R_i)
4     SI Oracle(S, C_i, ∇_i)
5        retourne VRAI
6     SINON
7        R_{i+1} ←Coupe(R_i, C_i, ∇_i)
8        i ← i + 1
9  retourne FAUX
```

4.6.2 Analyse de Complexité

Nous estimons tout d'abord le nombre d'itérations de notre méthode. Par la suite, nous analysons la complexité en temps de chaque itération.

Soit A_i l'aire du domaine de recherche R_i au début de l'itération i. L'algorithme applique une coupe valide passant par le centre de gravité C_i de R_i. Cette coupe est de la forme :

$$\nabla ep_S(C_i) \cdot v < \nabla ep_S(C_i) \cdot C_i \tag{4.20}$$

Soit R'_i le sous-ensemble de R_i correspondant aux points qui satisfont (4.20). D'après la proposition 8, l'aire de R'_i est inférieure à $(5/9)A_i$. Le domaine de recherche R_{i+1} à l'itération suivante correspond au plus grand polygone convexe à sommets portés par la grille inclus dans R'_i. Il est évident que l'aire de R_{i+1} est inférieure à l'aire de R'_i. Par conséquent, nous avons $A_{i+1} \leq (5/9)A_i$ et par extension nous avons : $A_i \leq (5/9)^i A_1$.

La figure 4.3 montre un exemple de coupe du domaine de recherche R_i passant par C_i. Nous vérifions bien que l'aire du domaine de recherche R_{i+1} est inférieure à 5/9 de l'aire du domaine R_i.

Lorsque l'aire du domaine de recherche est égale à zéro, cela signifie qu'il est réduit au polygone vide, à un simple point ou à un segment de droite. Si le domaine de recherche est réduit au polygone vide, l'ensemble de points S ne correspond pas à un morceau de plan discret naïf et l'algorithme se termine. Si le domaine de recherche est réduit à un seul point, le problème est résolu en temps linéaire puisqu'il suffit d'évaluer la fonction épaisseur en ce point restant. Dans le dernier cas, le centre de gravité correspond en fait au milieu du segment de droite. Le ratio associé à chaque coupe devient alors 1/2 et le problème est résolu en $\log_2 D$ itérations. Par conséquent, nous pouvons conclure que notre algorithme exécute $O(\log_{9/5} A_1)$ itérations dans le pire cas, où A_1 correspond à l'aire du domaine de recherche initial.

Proposition 9 *Le nombre d'itérations de notre algorithme admet une borne supérieure fonction de* A_1 *:*

$$\lfloor \log_{\frac{9}{5}} A_1 + \log_{\frac{9}{5}} 2 \rfloor + 3$$

Preuve 10 *Tant que le domaine de recherche R_i contient au moins trois points de la grille et qu'aucune solution n'a été trouvée, nous appliquons une coupe valide et l'aire du domaine de recherche R_{i+1} à l'itération suivante est inférieure à $(5/9)A_i$. Notons I_i le nombre de points de la grille contenus strictement à l'intérieur du domaine R_i. Notons B_i le nombre de points de la grille situés sur le bord de ce même domaine. Grâce au théorème de Pick (voir [Pic99] et la section 1.4.1) et comme R_i correspond à un polygone convexe à sommets portés par la grille, nous pouvons affirmer que $A_i = I_i + (B_i/2) - 1$. Par conséquent, l'inégalité suivante est toujours valide : $I_i + B_i \leq 2A_i + 2$. Pour déterminer le nombre maximum d'itérations k pour réduire le domaine de recherche à seulement deux points, nous devons donc simplement résoudre l'inéquation : $2A_k + 2 < 3$ équivalente à $2(5/9)^k A_1 < 3$. Il s'en suit que k est borné par $\log_{9/5} A_1 + \log_{9/5} 2$. Comme k est forcément une valeur entière, nous fixons cette borne supérieure à : $\lfloor \log_{9/5} A_1 + \log_{9/5} 2 \rfloor + 1$. Finalement, lorsque seuls deux points restent, nous les traitons en deux itérations au maximum. Nous remarquons que lorsque le domaine de recherche est réduit à un segment de droite alors le ratio de coupe devient $1/2$ ce qui n'augmente pas la borne supérieure pour le nombre d'itérations.*

Nous analysons à présent la complexité en temps de chaque itération. Le calcul du centre de gravité se fait en temps linéaire en fonction du nombre de sommets du domaine de recherche courant. Comme le nombre de coupes ne dépasse pas $O(\log D)$ et comme chaque coupe ajoute au plus $O(\log D)$ nouveaux sommets au domaine de recherche, le calcul du centre de gravité se fait en $O(\log^2 D)$. Par définition de la fonction épaisseur (définition 15), nous savons que l'évaluation de la fonction épaisseur en un point nécessite n produits scalaires. De plus, nous déduisons en temps constant de ce calcul un sous-gradient de la fonction épaisseur et nous connaissons donc la forme de la coupe valide à appliquer au domaine de recherche. La dernière étape consiste à reconstruire le domaine de recherche après la coupe. Pour cela, nous déterminons le plus grand polygone convexe à sommets portés par la grille inclus dans le domaine de recherche courant et dont les points satisfont l'inéquation de la coupe. Les sommets de ce polygone convexe sont situés dans une zone délimitée par la droite de coupe et deux côtés du domaine de recherche courant. Pour reconstruire le nouveau domaine de recherche, nous utilisons l'algorithme d'Harvey [Har99] ou notre méthode présentée dans la section 2.7.2 du chapitre 2 (voir aussi [CB09]). Ces deux approches s'exécutent en temps logarithmique par rapport aux coefficients de la normale de la droite de coupe et des droites portées par les deux côtés. Comme le vecteur normal de la droite de coupe correspond au sous-gradient, d'après (4.11) nous savons que ses coordonnées ne dépassent pas $O(D)$. De plus, en raison du choix du pas de discrétisation de la grille (voir Section 4.5), les coordonnées des deux autres vecteurs normaux ne dépassent pas $O(D^3)$. L'étape de reconstruction s'exécute donc en $O(\log D)$. En conclusion, comme $O(\log^2 D)$ et $O(\log D)$ peuvent être négligés par rapport à $O(n)$ (en considérant que $D \approx \sqrt{n}$), chaque coupe s'exécute en $O(n)$ dans le pire cas, ce qui implique que la complexité de notre méthode est $O(n \log D)$.

4.7 Algorithme Incrémental

Dans cette section, nous introduisons les motivations pour concevoir une version incrémentale de notre algorithme. Nous décrivons ensuite la méthode incrémentale et sa complexité.

4.7.1 Motivations et Description de l'Algorithme

En général, les données sur lesquelles nous travaillons correspondent à la dicrétisation d'objets réels. Ces données sont obtenues via des dispositifs numériques tels qu'un scanner 3D ou un télémètre laser. Après acquisition des données brutes, nous appliquons généralement des pré-traitements pour segmenter les données discrètes et sélectionner les sous-ensembles de points 3D qui sont susceptibles de correspondre à un morceau de plan discret. Ces pré-traitements utilisent généralement des critères géométriques au voisinage des points comme proposé dans [KBSS08] et dans [KBSS09]. Néanmoins, les méthodes utilisées pour discrétiser les objets continus ne fournissent pas des résultats exacts. En effet, la discrétisation d'un morceau de plan euclidien correspond généralement à un ensemble de points légèrement bombé. De plus, certains points en bordure de l'ensemble peuvent appartenir à une autre future facette de la polyédrisation de l'objet. Si nous lançons notre algorithme de reconnaissance de plans discrets sur un tel ensemble de points, il aura donc toutes les chances d'échouer. Dans ce cas, il est préférable d'exécuter l'algorithme de façon incrémentale. Pour cela, soit S l'ensemble de points tri-dimensionnel susceptible de correspondre à un morceau de plan discret, fourni par le pré-traitement. Soit S' un sous-ensemble de S ne contenant qu'un seul point. Nous ajoutons itérativement les autres points de S au sous-ensemble S', soit un par un, soit plusieurs à la fois, tant que S' correspond à un morceau de plan discret. Lorsque S' ne correspond plus à un morceau de plan discret, nous retirons le dernier point ajouté ou les derniers points ajoutés. Nous avons déterminé un sous-ensemble maximal de S correspondant à un morceau de plan discret et l'algorithme se termine. Une telle méthode est généralement appelée *croissance de région* et le premier point ajouté au sous-ensemble S' est le *point de départ* de la croissance.

Comme il y a de fortes chances pour que l'ensemble de point S soit un peu bombé, nous pouvons supposer qu'il est préférable de choisir comme point de départ le point au milieu de l'ensemble. Nous pouvons ensuite choisir les autres points à ajouter en utilisant les relations de voisinage entre les points (voir la section 1.1.1). Supposons que la projection des points de S sur le plan (x, y) est injective. À la première itération, nous ajoutons seulement le point de départ à S'. Ensuite, à chaque itération i, nous ajoutons à S' les 8-voisins des points ajoutés à l'itération précédente, soit un par un, soit tous à la fois.

4.7.2 Analyse de Complexité

La version incrémentale de notre méthode est très proche de la version non-incrémentale. Le pas de discrétisation du domaine de recherche est calculé de la même manière que dans la version non-incrémentale, en utilisant la taille d'une boite englobant l'ensemble de points S tout entier. Cependant, il n'est a priori pas possible d'initialiser le domaine de recherche par un rectangle comme décrit dans la section 4.5.2. En effet, nous ne connaissons pas les coordonnées des points qui seront ajoutés lors des itérations futures. Soit S' l'ensemble de points courant, nous appelons l'oracle en le centre de gravité de l'espace de recherche courant. Si l'oracle renvoie "faux", le domaine de recherche est réduit par application d'une coupe. Si l'oracle renvoie "vrai", un point ou une collection de points est ajouté au sous-ensemble S' que nous étudions, et l'oracle est rappelé en le même point de l'espace de recherche. Notons que dans ce cas, il est inutile de lancer l'oracle sur la totalité des points de l'ensemble S' courant. En effet, il est suffisant dans ce cas de ne tester que le ou les nouveaux points ajoutés juste avant l'appel à l'oracle ainsi que les deux points définissant le produit scalaire maximum et le produit scalaire minimum au dernier appel

de l'oracle. Lorsque le domaine de recherche est réduit à un polygone vide, nous retirons les derniers points ajoutés afin de sélectionner un ensemble de points qui correspond à un morceau de plan discret. La complexité de notre algorithme incrémental est donc en $O(n \log(D))$ dans le pire cas. En outre, nous remarquons que le fait d'ajouter les points un par un ou plusieurs à la fois ne modifie pas la complexité générale de la méthode.

L'algorithme 16 résume la version incrémentale de notre algorithme. En comparaison avec l'algorithme non-incrémental, nous avons besoin de deux fonctions supplémentaires : une pour l'ajout de points dans S', l'autre pour le retrait de points de S'. Remarquons que la fonction *ajoutePoints(S')* retourne une valeur booléenne pour indiquer si l'ensemble S' contient tous les points de S ou pas. De plus, à la fin de l'algorithme, l'ensemble de points S' correspond toujours à un morceau de plan discret. L'algorithme n'a donc plus aucune raison de retourner une valeur booléenne.

Algorithme 16 : Notre algorithme incrémental de reconnaissance de plans discrets

```
RECONNAISSANCE INCRÉMENTALE (S : ensemble de points 3D)
1  i ← 1    R_i ←InitialiseDomaineDeRecherche(S)
2  S' ← {pointMilieu(S)}     Plein ← FAUX
3  TANT QUE (NombreDeSommets(R_i) ≥ 1)
4     C_i ←CentreDeGravité(R_i)
5     SI (Oracle(S', C_i, ∇_i))
6        SI ( !Plein)
7           Plein ← ajoutePoints(S')
8        SINON
9           retourner
10    SINON
11       R_{i+1} ←Coupe(R_i, C_i, ∇_i)
12       i ← i + 1
13 RetirePoints(S')
```

4.8 Résultats Expérimentaux et Améliorations Pratiques

4.8.1 Résultats Expérimentaux

Comme l'algorithme le plus rapide que nous connaissons actuellement pour reconnaître un morceau de plan discret est l'algorithme des cordes (voir [GDRZ05]), nous allons comparer les temps d'exécution de notre méthode avec celui-ci. Pour obtenir les meilleurs temps possibles, nous testons les versions incrémentales et non-incrémentales de notre algorithme en procédant à une double coupe à chaque itération. Nous générons aléatoirement des ensembles de points tri-dimensionnels dont les coordonnées x et y sont denses dans un carré de côté $D - 1$. Nous nous assurons qu'ils correspondent à des morceaux de plans discrets naïfs. Pour créer des ensembles de points ne correspondant pas à des morceaux de plans discrets, nous modifions volontairement les coordonnées d'un point de l'ensemble. Nous testons les deux algorithmes avec ces ensembles de points. Comme l'algorithme des cordes calcule n produits scalaires à chaque itération tout comme notre méthode, nous pouvons comparer le nombre moyen d'itérations et les temps d'exécutions

	D	200	300	500	1000
Borne supérieure (itérations)	grille entière	62	67	72	79
	grille réduite	44	47	51	55
S = DP	Cordes (itérations)	10.15	10.22	10.91	11.35
	COBA (itérations)	8.24	8.56	9.44	10.10
	Cordes (temps)	0.514	1.155	3.429	14.267
	COBA (temps)	0.430	0.983	3.036	13.121
S = NDP	Cordes (itérations)	4.87	4.77	4.66	4.70
	COBA (itérations)	1.04	1.01	1.05	1.01
	Cordes (temps)	0.194	0.423	1.139	4.555
	COBA (temps)	0.056	0.116	0.330	1.257

TAB. 4.1 – Résultats expérimentaux quand S correspond à un plan discret (DP) ou pas (NDP), algorithmes non-incrémentaux

D	200	300	500	1000
cordes (itérations)	30.65	34.86	36.10	40.40
COBA (itérations)	16.21	18.25	19.55	21.80
cordes (temps)	0.274	0.640	1.330	5.742
COBA (temps)	0.237	0.521	1.286	5.272

TAB. 4.2 – Résultats expérimentaux quand S tout entier correspond à un plan discret, algorithmes incrémentaux

de ces deux méthodes. De plus, dans notre algorithme, comme les coordonnées des sommets du domaine de recherche ne peuvent pas toujours être représentées sur quatre octets, nous utilisons la bibliothèque gratuite GMP (GNU Multiple Precision Arithmetic Library) pour C++.

Les tableaux 4.1 et 4.2 montrent des résultats expérimentaux pour les versions incrémentales et non-incrémentales des deux algorithmes. Nous rappelons que $D-1$ correspond au plus grand côté d'une boite englobant l'ensemble de points S. Pour simplifier les tableaux, notre méthode est appelée COBA pour "Convex Optimization Based Algorithm". Grâce à la proposition 9, nous calculons la borne supérieure du nombre d'itérations de notre algorithme en fonction de D, dans le cas où nous partons du domaine de recherche réduit comme décrit dans la section 4.5.2 ou pas. Les résultats expérimentaux montrent qu'en moyenne notre algorithme effectue moins d'itérations et est plus efficace que l'algorithme des cordes lorsque l'ensemble de points est dense dans la boite englobante. De plus, notre méthode effectue environ cinq fois moins d'itérations en moyenne que la borne supérieure calculée. Nous remarquons également que notre algorithme incrémental est bien plus rapide que notre algorithme non-incrémental. Cela peut paraître surprenant mais ce phénomène s'explique de la manière suivante. Dans la version incrémentale, l'algorithme commence par appeler l'oracle pour des ensembles à faible nombre de points. Les premiers appels à l'oracle s'avèrent donc assez rapides tout en réduisant significativement l'espace de recherche par application de coupes. Par conséquent, l'oracle n'est appelé pour l'ensemble de points complet que très peu de fois, ce qui rend la méthode plus efficace en pratique.

4.8.2 Améliorations Pratiques

Pour obtenir de meilleurs temps d'exécution, nous pouvons ajouter deux améliorations pratiques à notre méthode.

Premièrement, nous remarquons que l'étape la plus lente de notre algorithme correspond à l'évaluation de la fonction épaisseur. Nous essayons donc d'accélérer cette phase de notre algorithme. En effet, cette étape consiste à calculer n produits scalaires afin de déterminer deux points p^a et p^b tels que le produit scalaire $C_i \cdot p^a$ est maximal et tel que le produit scalaire $C_i \cdot p^b$ est minimal. Ainsi, nous évaluons la fonction épaisseur et si celle-ci est supérieure à 1, nous calculons un sous-gradient pour générer les coupes. Nous montrons que, quand l'évaluation de la fonction épaisseur est supérieure à 1, il n'est pas toujours nécessaire de calculer n produits scalaires pour obtenir un sous-gradient. Soient p^c et p^d deux points de S tels que $C_i \cdot (p^c - p^d)$ est plus grand que 1. Soit ∇ le vecteur de la forme $proj_{\mathbb{R}^2}(p^c - p^d)$, ce vecteur correspond à un sous-gradient de la fonction épaisseur au point C_i. Par conséquent, les coupes de la forme $\nabla \cdot v < 1 - (p_3^c - p_3^d)$ et $-\nabla \cdot v < 1 + (p_3^c - p_3^d)$ correspondent à des coupes valides de notre problème de réalisabilité. Nous pouvons donc arrêter de calculer des produits scalaires dès que nous trouvons deux points p^c et p^d de S tels que $C_i \cdot (p^c - p^d)$ est supérieur à 1 et ensuite appliquer une double coupe valide sur le domaine de recherche. Bien sûr, ces coupes sont moins efficaces et donc le nombre d'itérations de notre méthode augmente. Cependant, le complexité en temps de notre algorithme est inchangée et les temps d'exécution sont significativement améliorés.

Deuxièmement, nous pouvons appliquer plus d'une double coupe à chaque itération en utilisant les points de S qui ont participé aux précédents calculs de sous-gradient. Soient p^{ai} et p^{bi} les deux points de S tels que le sous-gradient à l'itération i soit de la forme $proj_{\mathbb{R}^2}(p^{ai} - p^{bi})$. À l'itération k, nous pouvons appliquer en plus de la double coupe classique $k-1$ doubles "coupes croisées" en utilisant les couples de points p^{ak} et p^{bi} avec $1 \leq i < k$ ainsi que $k-1$ doubles coupes croisées par rapport aux couples de points p^{ai} et p^{bk} avec $1 \leq i < k$. Comme chaque double coupe s'exécute en $O(\log D)$ et comme nous pouvons considérer que $O(\log^2 D)$ peut être négligé par rapport à $O(n)$ (en considérant que $D \approx \sqrt{n}$), ce changement ne modifie pas la complexité de l'algorithme. Cependant, cela améliore légèrement son temps d'exécution.

Le tableau 4.3 montre nos résultats expérimentaux lorsque nous ajoutons ces deux améliorations pratiques : moins de tests durant l'oracle et les coupes croisées. Nous mettons également en évidence le pourcentage de points testés durant l'oracle en mettant en oeuvre la première amélioration. Pour mémoire, nous rappelons également les résultats expérimentaux de notre algorithme sans aucune des deux améliorations pratiques. Nous remarquons que ces modifications accélèrent l'exécution de l'algorithme. Par exemple, lorsque $D = 1000$, notre méthode s'exécute en environ 6.245 secondes avec les deux améliorations, contre 13.121 secondes en moyenne sans ces deux améliorations.

4.9 Conclusion

Nous décrivons une nouvelle approche pour la reconnaissance de plans discrets naïfs. Nous montrons comment transformer ce problème de reconnaissance tri-dimensionnel en un problème de réalisabilité sur une fonction convexe bi-dimensionnelle. Cette fonction convexe calcule la distance verticale entre deux plans parallèles supports encapsulant l'en-

	D	200	300	500	1000
Sans les améliorations	itérations	8.24	8.56	9.44	10.10
	temps	0.430	0.983	3.036	13.121
Moins de Tests durant Oracle (MTO)	itérations	18.24	20.07	22.38	25.80
	temps	0.302	0.670	1.666	7.372
MTO et coupes croisées	itérations	15.87	16.83	19.02	21.20
	temps	0.330	0.632	1.673	6.245
Points testés		21.4%	20.3%	16.4%	16.2%

Tab. 4.3 – Résultats expérimentaux de COBA avec améliorations pratiques

semble de points et dont les coordonnées du vecteur normal correspondent aux paramètres donnés à la fonction. Nous montrons comment évaluer cette fonction en un point et comment en déduire un sous-gradient en temps linéaire. Comme le domaine de recherche est planaire et discrétisé, nous appliquons des techniques géométriques simples pour réduire le domaine de recherche jusqu'à trouver une solution au problème de réalisabilité. De plus, nous choisissons de réduire le domaine en appliquant des coupes particulières passant par son centre de gravité ce qui assure un nombre logarithmique d'itérations. Cette version de notre algorithme atteint une complexité dans le pire cas en $O(n \log D)$. Nous présentons différentes améliorations pratiques qui accélèrent l'exécution de notre méthode. Les résultats expérimentaux que nous obtenons montrent que notre algorithme reconnaît un morceau de plan discret naïf de 10^6 voxels en 10 itérations linéaires en moyenne, ou en environ 6 secondes avec les améliorations. Ces résultats impliquent que notre algorithme est plus efficace en moyenne que l'algorithme des cordes lorsque l'ensemble de points à traiter est dense. Notre méthode a été conçue pour être efficace en pratique en particulier pour des ensembles de points denses dans une boite englobante. Il s'agit de la seule méthode connue des auteurs qui atteint une complexité de $O(n \log D)$ dans le pire cas et qui est efficace en pratique. Le problème de reconnaissance dans $\mathbb{Z}^d$ peut toujours être transformé en un problème de réalisabilité dans $\mathbb{Z}^{d-1}$, quelle que soit la dimension. Néanmoins, certaines étapes de notre méthode utilisent des algorithmes géométriques planaires dont l'extension en dimensions supérieures n'est pas évidente. C'est pourquoi, à ce jour, notre algorithme n'est valide que pour reconnaître des ensembles de points tri-dimensionnels. En outre, d'autres améliorations pratiques peuvent probablement être apportées pour accélérer encore l'exécution de la méthode.