Nuages de Points et Modélisation 3D

Rapport de projet : Screened Poisson Surface Reconstruction

Jean Feydy
École Normale Supérieure

Table des matières

1 Introduction

Le schéma de reconstruction de surface dit *de Poisson*, proposé par Kazhdan, Bolitho et Hoppe en 2006 [1], est désormais un classique, dont nous avons brièvement parlé en cours. Publié sept ans plus tard par les mêmes auteurs, [2] améliore le procédé dans deux grandes directions : en contraignant la surface à passer au plus près des points de données, il permet une interpolation plus fine et moins lissée du nuage de point ; par l'utilisation efficace d'un Octree, il permet de passer d'une complexité log-linéaire à une complexité linéaire en le nombre de points du nuage.

Après un bref rappel sur la méthode de Poisson, nous exposerons la manière dont la surface reconstruite est contrainte à passer au mieux près des détails du nuage, par l'ajout d'un terme d'attache aux données. Nous décrirons alors notre implémentation de l'algorithme en deux dimensions qui, partant d'un nuage de point munis de normales orientés, reconstruit une surface – ou, dans notre cas, une ligne – en utilisant les idées de [1] et [2].

2 Formulation mathématique

L'idée centrale de la méthode de Poisson est que, si $\chi : \mathbb{R}^3 \to \mathbb{R}$ est la fonction indicatrice de notre solide $\mathbb{1}_V$, ou, de manière plus symétrique, $\mathbb{1}_V - 1/2$, alors le gradient de f peut s'identifier avec le champ de normales entrantes à V. Étant donné un nuage de points orientés P, il semble donc pertinent d'interpoler sur $\mathbb{R}^3$ tout entier un champ de vecteur $\vec{V}$, approchant au mieux le champ de gradient d'une version lissée de l'indicatrice $\mathbb{1}_V$: le calcul, complexe, est traité en détail section 3.

Mais notre champ de vecteur interpolé n'est pas, a priori, un champ de gradient, vérifiant l'identité fondamentale :

$$\int_\gamma \vec{\nabla} X \cdot \mathrm{d}\vec{l} = X(B) - X(A), \tag{1}$$

pour tout chemin γ C^1 reliant A à B.

Pour retrouver V, puis sa frontière, on se résoud donc à chercher un potentiel χ dont le gradient est le plus proche possible de $\vec{V}$ en norme L^2, c'est à dire qui minimise l'énergie

$$E_{\vec{V}}(\chi) = \int \left\| \nabla\chi(p) - \vec{V}(p) \right\|^2 \mathrm{d}p. \tag{2}$$

Il suffit alors de résoudre l'équation de Poisson vérifiée aux points critique de $E_{\vec{V}}$,

$$\Delta\chi = \nabla \cdot \vec{V}, \tag{3}$$

avec des conditions aux bord pertinentes (χ constante égale à $-1/2$ sur le bord, par exemple, dans le cas d'une surface fermée) puis de définir notre surface reconstruite S comme l'ensemble des points d'annulation de χ.

Malheureusement, les solutions obtenues par ce procédé ont une fâcheuse tendance à être trop lisses, à ne pas bien saisir les détails, les textures de l'objet numérisé.

L'idée majeure de [2] est alors d'introduire un terme d'attache aux données P,

$$E_{(w,P)}(\chi) = \frac{\text{Aire}(P)}{\sum_{p\in P} w(p)} \sum_{p\in P} w(p)\chi^2(p), \qquad (4)$$

avec $w : P \to \mathbb{R}_+$ un poid, relatif à la confiance accordée à chaque mesure. On voit ici tout l'intérêt d'avoir pris comme convention $\chi = 0$ sur la surface reconstruite, tandis que l'aire de la surface initiale, estimée via un calcul de la densité de l'échantillonnage, sert ici de facteur d'échelle. L'énergie à minimiser devient alors, avec α un paramètre réel,

$$E(\chi) = E_{\vec{V}}(\chi) + \alpha E_{(w,P)}(\chi). \qquad (5)$$

Quitte à recentrer et à changer d'échelle, on peut supposer travailler sur le cube $[0,1]^3$, et l'on peut alors réécrire notre énergie,

$$E(\chi) = \langle \vec{V} - \nabla\chi, \vec{V} - \nabla\chi \rangle_{[0,1]^3} + \alpha \langle \chi, \chi \rangle_{(w,P)}, \qquad (6)$$

avec

$$\langle \vec{U}, \vec{V} \rangle_{[0,1]^3} = \iiint \langle \vec{U}(x), \vec{V}(x) \rangle \mathrm{d}x, \qquad (7)$$

$$\langle F, G \rangle_{(w,P)} = \frac{\text{Aire}(P)}{\sum_{p\in P} w(p)} \sum_{p\in P} w(p)F(p)G(p). \qquad (8)$$

La minimisation effective est abordée dans la section 4, mais, avant d'entrer dans les détails du calcul de $\vec{V}$ et χ, nous souhaiterions souligner l'intérêt crucial de l'utilisation d'un Octree dans la résolution de notre problème.

2.1 Utilisation d'un octree

Pour résoudre un problème de Poisson comme celui présenté plus haut, on utilise d'ordinaire un maillage, plus ou moins raffiné selon le niveau de détails recherché. Dans notre cas, une discrétisation uniforme de l'espace $[0,1]^3$ n'est guère appropriée : si l'on cherche à garantir un niveau de détails élevé dans les zones ou la densité de points est la plus forte, il est superflu d'engager d'importantes ressources pour le reste de l'espace, vide de point, qui ne nous intéresse qu'indirectement. Nous avons donc besoin d'un maillage d'autant plus fin que la densité des données est forte.

Une manière efficace d'obtenir une telle discrétisation de l'espace est d'utiliser un *Octree* : après avoir correctement choisi notre système de coordonnées pour que notre nuage soit, avec une certaine marge, compris dans le cube $[0,1]^3$, on construit un Octree volumique T d'une profondeur arbitraire, de façon à ce que chaque feuille ne contienne au plus qu'un point du nuage P. Bien entendu, dans la pratique, on se limite à une profondeur maximale D, qui n'est atteinte que pour des écarts inter-points de l'ordre de 2^{-D} : un exemple de tel arbre, en deux dimensions, est donné Figure 1.

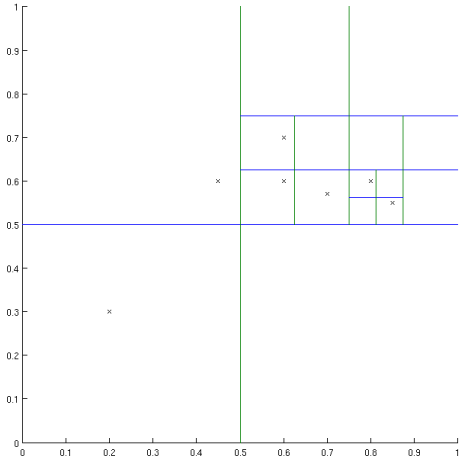


FIGURE 1 – Un exemple de Quadtree – équivalent en deux dimensions d'un Octree – construit sur le carré unité à partir des points noirs.

3 Estimation du champ de vecteurs

3.1 Recherche d'une formule pertinente

La première étape du calcul, traitée en détail dans [1], est l'interpolation du champ $\vec{V}$ à partir de données isolées, le champ de normales entrantes

$\vec{N}_p$, pour $p \in P$.

Un premier calcul, à densité constante On ne cherchera pas ici à calculer exactement le champ de normale entrante, qui n'est supporté que par ∂V, mais plutôt une approximation lissée. Donnons donc une fonction de flou, F, positive centrée d'intégrale 1 – typiquement, une gaussienne d'écart-type $d/2$, où d est la distance moyenne d'échantillonnage sur la surface.

Il serait alors tentant de définir notre champ $\vec{V}$ par la formule :

$$\vec{V}(q) = \sum_{p \in P} F(q-p)\vec{N}_p, \tag{9}$$

somme de "gaussiennes vectorielles" centrées sur les points du nuage P.

Écriture sur l'octree Pour améliorer les performances de notre algorithme, nous rangeons nos points dans un octree, défini de sorte que chacune de ses feuilles ne contienne qu'au plus un point de P. En supposant notre nuage de points échantillonné régulièrement, avec une distance de l'ordre de d entre chaque point, on peut en bonne approximation supposer que nos points sont tous tenus séparés dans des feuilles de profondeur $D = \text{ceil}(-\log_2(d))$. Aussi, plutôt que de garder nos petites gaussiennes centrées sur les points du nuage, on préfère – pour des raisons d'efficacité – les répartir sur les feuilles de profondeur D de l'octree. Avec F_d gaussienne centrée d'écart-type $d/2$, et en exprimant tout point p du nuage comme le barycentre trilinéaire des huit centres de feuilles les plus proches, i.e.

$$p = \sum_{f \in \text{Vois}_D(p)} \alpha_{f,p} f, \tag{10}$$

on définit $\vec{V}$ par :

$$\vec{V}(q) = \sum_{p \in P} \sum_{f \in \text{Vois}_D(p)} \alpha_{f,p} F_d(q-f)\vec{N}_p \tag{11}$$

$$= \sum_{f \in \text{Vois}_D(P)} F_d(q-f) \cdot \left(\sum_{p \in \text{Infl}(f) \cap P} \alpha_{f,p}\vec{N}_p \right), \tag{12}$$

avec $\text{Infl}(f)$, la zone d'influence de la feuille f, le cube centré sur f de côté d.

Une formule adaptée aux densités variables Une formule étant maintenant établie dans le cas où les points sont régulièrement échantillonnés, on l'étend au cas où elle varie. Une fois encore, l'idée est simple : un point du nuage comptant pour un petit élément de surface, la contribution d'un point au champ $\vec{V}$ sera proportionnelle au carré de la distance qui le sépare de ses plus proches voisins, *inversement proportionnelle* à la densité locale au point considéré.

Il importe donc de savoir estimer la densité locale de l'échantillonnage W en un point du nuage. On définit donc, pour $\tilde{D}$ une profondeur inférieure à D la profondeur maximale de l'arbre :

$$W_{\tilde{D}}(q) = \sum_{p \in P} \sum_{n \in \text{Vois}_{\tilde{D}}(p)} \alpha_{n,p} F_n(q) \tag{13}$$

$$\simeq \sum_{p \in P} F_{2^{-\tilde{D}}}(q-p), \tag{14}$$

où F_n, fonction attachée au noeud n, est une gaussienne centrée sur n d'écart-type $2^{-\tilde{D}}/2$. Prendre $W_{\tilde{D}}$ comme estimateur de la densité revient donc à diffuser l'influence de chacun de nos points à une échelle $2^{-\tilde{D}}$.

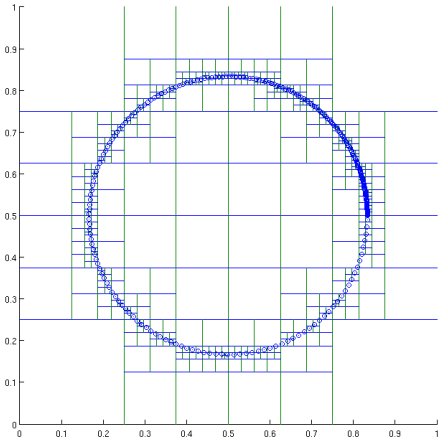
Ainsi, une formule un peu plus pertinente pour $\vec{V}$ est elle, avec D la profondeur maximale de l'arbre – profondeur d'estimation de $\vec{V}$ – et $\tilde{D}$ profondeur d'estimation de la densité,

$$\vec{V} = \sum_{p \in P} \frac{1}{W_{\tilde{D}}(p)} \sum_{f \in \text{Vois}_D(p)} \alpha_{f,p} F_d(q-f)\vec{N}_p. \tag{15}$$

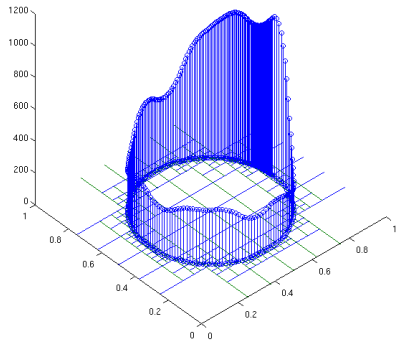
Échelle de diffusion variable Une dernière faiblesse toutefois : si un point p se trouve dans une zone très peu échantillonnée, diffuser sa normale $\vec{N}_p$ à une échelle aussi fine que 2^{-D} fait peu de sens, et mène à un champ $\vec{V}$ "à trous" : voir Figure 3b. On définit donc la profondeur de diffusion, $\text{Prof}(p)$, par

$$\text{Prof}(p) = \min(D, D + \log_4(W_{\tilde{D}}(p)/W)) \tag{16}$$

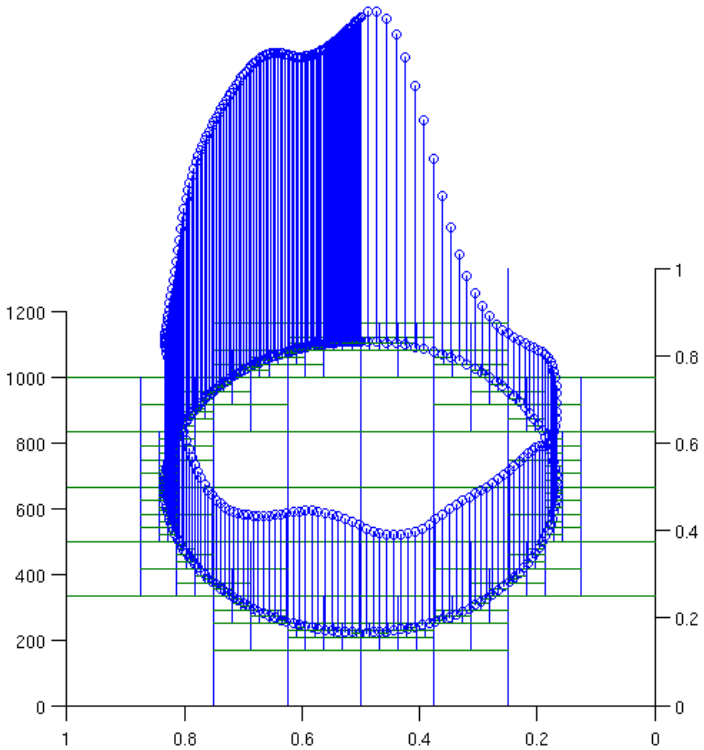
où W est la valeur moyenne de $W_{\tilde{D}}$ sur le nuage P. Cette formule et son $\log_4$ sont bien pertinentes : pour qu'un point opère sur un rayon deux fois supérieur, il faut qu'il compte, littéralement, pour 2^2 points "moyens" (nous nous trouvons en bonne approximation sur une surface, de dimension 2). Il



(a) Vue de la répartition des points.



(b) Les valeurs prises par W.



(c) Le lissage de W au point d'angle $\theta = 0$ correspond bien à la taille du support de la fonction de lissage utilisée, présentée plus loin.

FIGURE 2 – Quelques vues de l'estimateur de densité W, calculé à profondeur 3, sur une série de points disposés le long d'un cercle, avec une densité d'échantillonnage fonction affine décroissante de l'angle. L'anisotropie du QuadTree induit les faibles oscillations le long du cercle.

est donc normal d'exiger que la densité y soit 4 fois inférieure à sa valeur moyenne.

Ainsi, pour notre implémentation 2D de l'algorithme, le $\log_4$ est-il remplaçé par un $\log_{2^1}$: les points sont supposés se trouver sur une courbe, de dimension 1.

On obtient, donc, pour finir cette section d'une formule permettant d'interpoler nos normales rentrantes $\vec{N}_p$ de manière convaincante :

$$\vec{V}(q) = \sum_{p \in P} \frac{1}{W_{\tilde{D}}(p)} \sum_{n \in \text{Vois}_{\text{Prof}(p)}} \alpha_{n,p} F_n(q) \vec{N}_p, \quad (17)$$

avec F_n fonction positive d'intégrale 1 diffusant à l'échelle du noeud n autour de celui-ci. Le paramètre $\tilde{D}$, quand à lui, devrait être pris de façon à ce que 2^{-D} soit de l'ordre de la longueur caractéristique d'évolution de la densité d'échantillonnage.

3.2 Implémentation

Le point clé est l'utilisation de l'octree dans l'expression du champ, qui est simplement encodé par la donné, en chaque noeud de l'arbre, d'un vecteur d'accumulation des $\vec{N}_p$ correctement balancés. Aussi, l'étape de calcul de ces vecteurs d'accumulation – `T.loadV()` – est-elle codée comme suit :

- Dans un premier temps, on calcule la densité de l'échantillonnage en chaque point, $W_{\tilde{D}}(p)$.
- On en déduit ensuite les profondeurs de diffusion $\text{Prof}(p)$.
- Enfin, pour chaque point du nuage p, on diffuse sa normale $\vec{N}_p$ à une échelle $2^{-\text{Prof}(p)}$ dans un voisinage dépendant du support de la fonction F.

Choix de la fonction de référence F En terme de complexité, il est donc crucial que la fonction F soit à support compact, pour passer d'une complexité quadratique à log-linéaire en le nombre de points de P.

Malheureusement, les gaussiennes ne sont pas à support compact...On choisi donc de prendre pour F une des autoconvolées $B_n = (1_{[-1/2,1/2]^3})^{\star n}$, qui présentent de nombreux avantages. La séparation des variables, tout d'abord, qui permet d'écrire

$$B_n(x,y,z) = f_n(x) f_n(y) f_n(z) \quad (18)$$

où $f_n = (1_{[-1/2,1/2]})^{\star n}$ est une fonction polynomiale par morceaux, d'ordre $n-1$, positive, d'intégrale 1 et à support compact $[-n/2, n/2]$. Le théorème central limite assure de plus que nos fonctions B_n convergent faiblement vers une gaussienne centrée, ce qui est rassurant – en pratique, on se limite au cas $n = 3$, amplement suffisant. La fonction F_n, associée au noeud de centre c, de côté l, est alors définie par

$$F_n(q) = F((q-c)/l)/l^3. \quad (19)$$

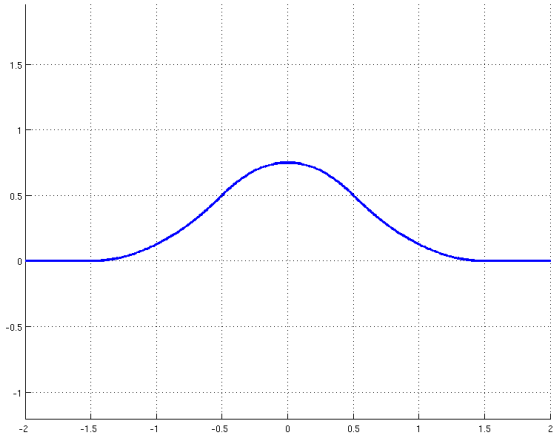


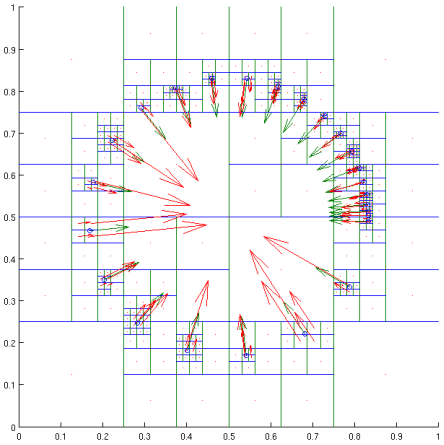
FIGURE 4 – La fonction f_3, spline élémentaire d'ordre 2.

4 Méthode des Éléments Finis en cascade sur l'Octree

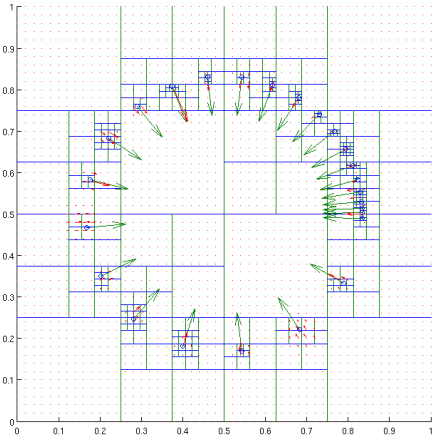
4.1 Méthode des éléments finis

On fait ici le choix d'utiliser une approche variationnelle de la méthode des éléments finis. Étant donnée une base de fonctions $(B_i : [0,1]^3 \to \mathbb{R})_{i \in [\![1,N]\!]}$, on cherche χ sous la forme

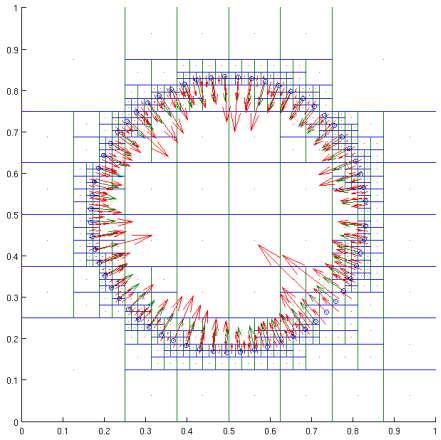
$$\chi(p) = \sum_{i=1}^{N} \lambda_i B_i(p), \quad (20)$$



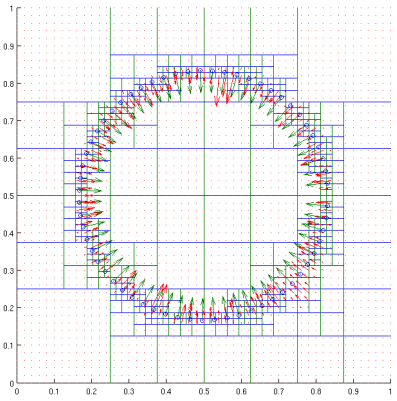
(a) Si la variabilité des densités est trop grande, notre formule pour Prof(p) perd en pertinence.



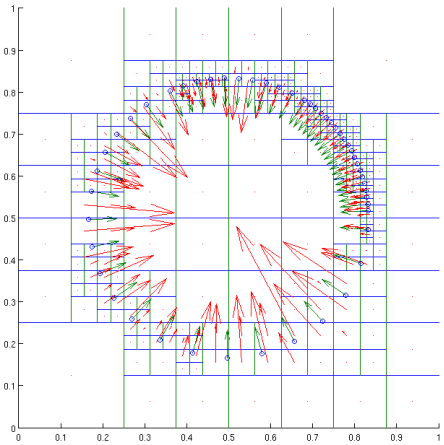
(b) Le champ $\vec{V}$ est alors “à trous”.



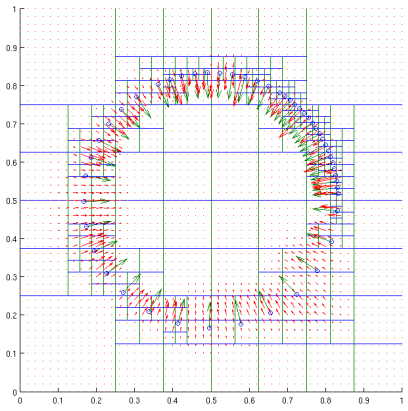
(c) À l’inverse, si la densité d’échantillonnage est uniforme...



(d) Le champ reconstruit est bien uniformément radial.



(e) Enfin, si la densité varie sans sauter plusieurs ordres de grandeur...



(f) Le résultat est conforme à nos attentes.

FIGURE 3 – Exemples de champs $\vec{V}$ calculés sur un cercle.

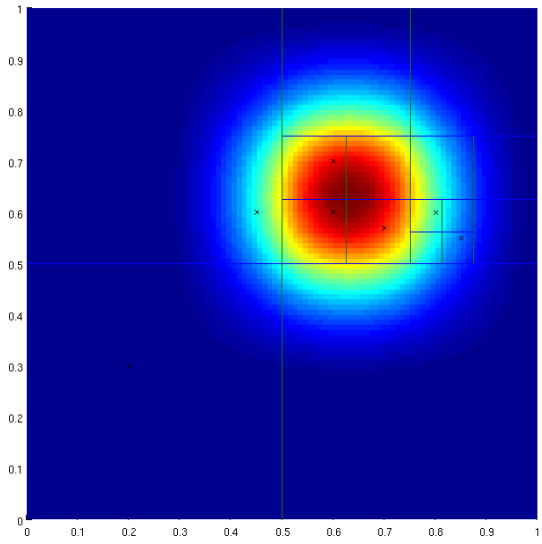


FIGURE 5 – La fonction associée au noed $[4, 1]$.

et l'on minimise, en fonction de la variable $\lambda \in \mathbb{R}^N$, l'énergie

$$E(\lambda) = \langle \vec{V} - \nabla \sum \lambda_i B_i, \vec{V} - \nabla \sum \lambda_i B_i \rangle_{[0,1]^3} \tag{21}$$

$$+ \alpha \cdot \langle \sum \lambda_i B_i, \sum \lambda_i B_i \rangle_{(w,P)}. \tag{22}$$

La bilinéarité de nos deux produits $\langle \cdot, \cdot \rangle_{[0,1]^3}$ et $\langle \cdot, \cdot \rangle_{(w,P)}$ nous permet alors de développer

$$E(\lambda) = \langle \vec{V}, \vec{V} \rangle \tag{23}$$

$$- 2 \cdot \lambda^T \left(\langle \vec{V}, \nabla B_i \rangle \right)_i \tag{24}$$

$$+ \lambda^T (\langle \nabla B_i, \nabla B_j \rangle)_{i,j} \lambda \tag{25}$$

$$+ \alpha \cdot \lambda^T \left(\langle B_i, B_j \rangle_{(w,P)} \right)_{i,j} \lambda. \tag{26}$$

À l'optimum, on a donc

$$A\lambda = b, \tag{27}$$

avec

$$A_{i,j} = \langle \nabla B_i, \nabla B_j \rangle + \alpha \cdot \langle B_i, B_j \rangle_{(w,P)}, \tag{28}$$

$$b_i = \langle \vec{V}, \nabla B_i \rangle. \tag{29}$$

La matrice A étant symétrique définie positive dès que w est non-nulle sur au moins deux points distincts, le problème est résolu, modulo le choix des B_i – et une implémentation efficace !

4.2 Choix de la base de fonctions et calcul des coefficients matriciels

C'est tout naturellement que l'on reprend pour fonctions de base les fonctions associées, section 3, aux noeuds de notre Octree : simples à utiliser, elles permettent d'exprimer avec finesse nos solutions au voisinage de la surface, et sans overfitting dans les régions vides de points. Reste alors à calculer les coefficients b_i et $A_{i,j}$: c'est ici que l'on mesure tout l'intérêt d'avoir paramétré $\vec{V}$ par des vecteurs attachés aux noeuds de l'arbre, et représenté $\vec{V}$ comme une somme de fonctions nodales. En dehors du calcul des $\langle B_i, B_j \rangle_{(w,P)}$, rendu simple et efficace grâce à l'organisation spatiale en octree des points du nuage, on peut rammener le calcul de nos coefficients à des calculs de $\langle \nabla G, \nabla H \rangle$ et $\langle G, \nabla H \rangle$ où G et H sont des fonctions nodales, polynomiales par morceaux. Examinons de plus près notre algorithme :

Calcul des coefficient de b Pour calculer l'influence de $\vec{V}$ sur notre système, rappellons l'expression de $\vec{V}$ adoptée section 3 :

$$\vec{V}(q) = \sum_{p \in P} \frac{1}{W_{\tilde{D}}(p)} \sum_{n \in \text{Vois}_{\text{Prof}(p)}} \alpha_{n,p} F_n(q) \vec{N}_p \tag{30}$$

$$= \sum_{n \in A} F_n(q) \vec{V}_n, \tag{31}$$

avec A notre arbre, et $\vec{V}_n$ vecteur d'accumulation au noeud n. Aussi, avec $B_i = F_m$ fonction associée au noeud m, on a

$$b_i = \langle \vec{V}, \nabla B_i \rangle \tag{32}$$

$$= \langle \sum_{n \in A} F_n \vec{V}_n, \nabla F_m \rangle \tag{33}$$

$$= \sum_{n \in \text{Interf}(m)} \langle F_n \vec{V}_n, \nabla F_m \rangle \tag{34}$$

$$\tag{35}$$

où $\text{Interf}(m)$ est l'ensemble des noeuds de A qui interfèrent avec m, i.e. tels que le support de F_n intersecte celui de F_m. Plus simplement, cela signifie que la distance en norme infinie entre les deux centres c_n et c_m est inférieure à une fois et demie la somme des deux côtés des noeuds l_n et l_m. Dans notre implémentation, nous avons simplement utilisé le fait que, si n n'interfère pas avec m, alors ses

enfants non plus : un simple parcours en profondeur de l'arbre, avec une descente stoppée lorsque l'on cesse d'interférer avec m, suffit à obtenir une complexité raisonnable.

On utilise alors le fait que

$$\begin{aligned}\langle F_n \vec{V}_n, \nabla F_m\rangle &= \vec{V}_{n,x}\langle F_n, \partial_x F_m\rangle && (36)\\ &+ \vec{V}_{n,y}\langle F_n, \partial_y F_m\rangle && (37)\\ &+ \vec{V}_{n,z}\langle F_n, \partial_z F_m\rangle, && (38)\end{aligned}$$

puis

$$\begin{aligned}\langle F_n, \partial_x F_m\rangle &= \langle \frac{F((\cdot - c_n)/l_n)}{l_n^3}, \partial_x \frac{F((\cdot - c_m)/l_m)}{l_m^3}\rangle && (39)\\ &= \frac{1}{l_n^3 l_m^{3+1}}\langle F((\cdot - c_n)/l_n), (\partial_x F)((\cdot - c_m)/l_m)\rangle && (40)\\ &= \frac{1}{l_n^3 l_m^{3+1}}\langle F((\cdot - (c_n - c_m))/l_n), (\partial_x F)(\cdot/l_m)\rangle && (41)\\ &= \frac{1}{l_n^3 l_m^{1}}\langle F((\cdot\ l_m - (c_n - c_m))/l_n), (\partial_x F)(\cdot)\rangle && (42)\\ &= \frac{l_m^3}{l_n^3 l_m^{3+1}}\langle F((\cdot - (c_n - c_m)/l_m)/(l_n/l_m)), (\partial_x F)(\cdot)\rangle && (43)\\ &= \frac{1}{l_m^4}\langle F_{\tilde{n}}, \partial_x F\rangle, && (44)\end{aligned}$$

avec $\tilde{n}$ le noeud de centre $(c_n - c_m)/l_m$, de côté l_n/l_m, et ce, modulo les problèmes de domaine d'intégration, omis pour des raisons de lisibilité, mais bien présent, car notre produit scalaire est défini comme étant l'intégration sur le seul cube unité. On utilise alors le fait que $\partial_x F$ est polynomiale par morceaux, bien connue, et que $F_{\tilde{n}}$ est polynomiale par morceaux, aux coefficients faciles à calculer, pour calculer explicitement la valeur du produit voulue – la séparabilité de F permettant bien entendu de simplifier considérablement les calculs.

Calcul des coefficient de A De même, on souhaite calculer de manière efficace $\langle \nabla F_n, \nabla F_m\rangle$ pour n et m deux noeuds interférents. Pour ce faire, il est souhaitable de se rammener, par un changement de variable, à un produit du type $\langle \nabla F_{\tilde{n}}, \nabla F\rangle$, facile à implémenter. On écrit donc :

$$\begin{aligned}\langle \nabla F_n, \nabla F_m\rangle &= \langle \partial_x F_n, \partial_x F_m\rangle && (45)\\ &+ \langle \partial_y F_n, \partial_y F_m\rangle && (46)\\ &+ \langle \partial_z F_n, \partial_z F_m\rangle, && (47)\end{aligned}$$

puis

$$\begin{aligned}\langle \partial_x F_n, \partial_x F_m\rangle &= \langle \partial_x \frac{F((\cdot - c_n)/l_n)}{l_n^3}, \partial_x \frac{F((\cdot - c_m)/l_m)}{l_m^3}\rangle && (48)\\ &= \frac{1}{l_m^{3+1}}\langle \partial_x \frac{F((\cdot - (c_n - c_m))/l_n)}{l_n^3}, (\partial_x F)(\cdot/l_m)\rangle && (49)\\ &= \frac{1}{l_m^{1}}\langle \partial_x \frac{F((\cdot\ l_m - (c_n - c_m))/l_n)}{l_n^3}, \partial_x F(\cdot)\rangle && (50)\\ &= \frac{1}{l_m^{1}}\langle \partial_x \frac{F((\cdot\ l_m - (c_n - c_m))/l_n)}{l_n^3}, \partial_x F(\cdot)\rangle && (51)\\ &= \frac{1}{l_m^{3+1}}\langle \partial_x \frac{F((\cdot - (c_n - c_m)/l_m)/(l_n/l_m))}{(l_n/l_m)^3}, \partial_x F(\cdot)\rangle && (52)\\ &= \frac{1}{l_m^{3+1}}\langle \partial_x F_{\tilde{n}}, \partial_x F\rangle, && (53)\end{aligned}$$

avec $\tilde{n}$ le noeud de centre $(c_n - c_m)/l_m$, de côté l_n/l_m. Un tel produit se calcule donc en un coût constant.

Pour calculer A toute entière en un temps raisonnable, il suffit ensuite de constater deux choses :

- A est symétrique, ce qui permet de limiter les calculs aux couples $\langle \nabla F_n, \nabla F_m\rangle$ avec $d_n \geqslant d_m$, où d_n, d_m profondeur respectives des noeuds n et m.
- Si $d_n > d_m$, il y a au plus 4^3 noeuds de profondeur d_n interférant avec m, et il est facile de les obtenir – voir `QuadTree.overlappingNodesAtDepthD`.
  Si $d_n = d_m$, ce nombre passe à 5^3.

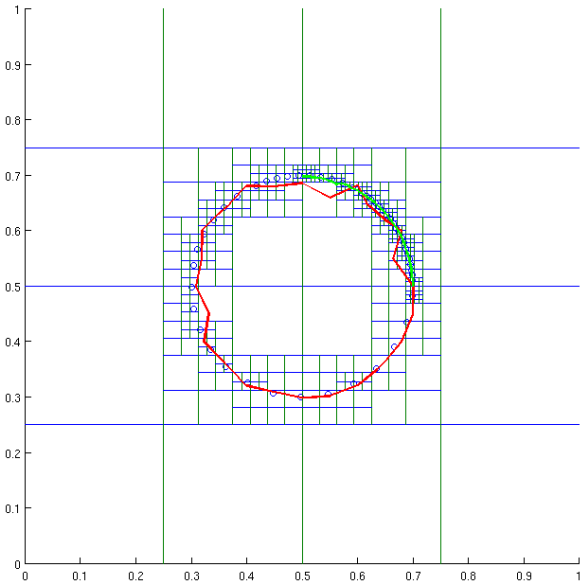
Aussi, en parcourant les lignes de la matrice A, et en ne calculant que les coefficients correspondant aux noeuds possiblement interférant, on obtient une complexité log-linéaire pour le calcul de A.

Nous implémentons, pour finir, l'Algorithme 1 de l'article [2], suivant une approche coarse-to-fine classique, mais en augmentant le poid d'attache aux données α à mesure que la profondeur augmente, afin de conserver un équilibre entre le terme de laplacien et le terme d'attache aux données, entre l'énergie $E_{\vec{V}}(\chi)$ et $\alpha E_{(w,P)}(\chi)$.

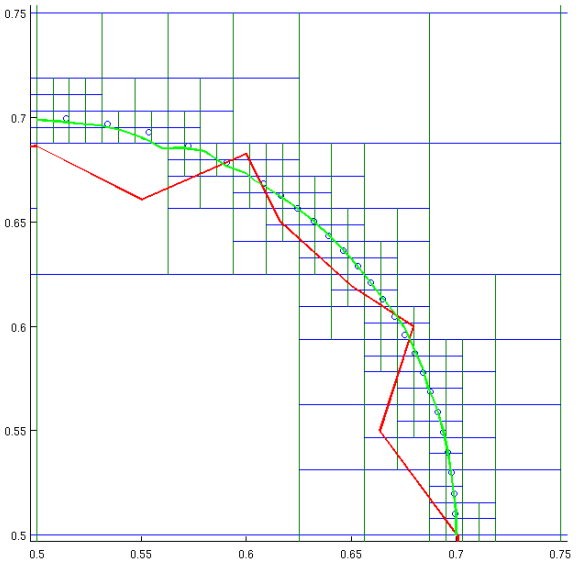
5 Étude des résultats et travail restant à accomplir

Pour finir, nous essayons notre code sur un exemple simple, obtenu en échantillonnant un cercle avec une densité variable. Si les premiers essais semblent décevants, cela est dû à l'étape de tracé de contour : sans implémentation spécifique, utiliser la fonction `contour` de Matlab (qui semble opérer un marching cubes) nécessite l'utilisation d'une grille de taille fine. La Figure 6 laisse à penser que nous n'avons pas fait de grosse erreur de calcul ou d'implémentation, ce qui est rassurant.

Au terme de ce projet, nous avons donc réussi à mettre en application, de manière sommaire, les principales idées développées dans [1] et [2], en deux dimensions – l'extension à la 3D ne poserait guère de difficulté, mais les performances de notre code Matlab étant relativement médiocres, insister dans cette direction ne nous a pas semblé très judicieux. Trois idées améliorant les performances restent néanmoins à implémenter : le clustering des points du nuage aux grandes échelles ; l'utilisation d'un arbre conforme pour ouvrir la voie à une résolution linéaire en le nombre de points ; l'obtention d'un mesh par un marching cubes sur l'octree – et non plus sur une grille, comme le fait Matlab par défaut. Le problème des conditions aux bords du carré unité ayant été ici ignoré, il serait aussi bon d'en faire un traitement plus complet par la suite, ainsi que de tester notre implémentation sur des cas plus ardus qu'un simple cercle.



(a) Deux visualisations de la même isosurface...



(b) Comme on le voit sur ce zoom, le résultat vert, obtenu avec un marching cubes fin, est bien meilleur que ce qu'un marching cubes lâche laissait espérer.

FIGURE 6 – Illustration de notre algorithme sur un exemple circulaire simple.

Références

[1] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[2] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3) :29, 2013.