

Interpolation de Sibson

Interpolation par voisins naturels

Tong Zhao, Pierre Boyeau

Mai 2017

Ecole des Ponts et Chaussées

1. Principe théorique et propriétés
2. Implémentation numérique
3. Conclusion

Principe théorique et propriétés

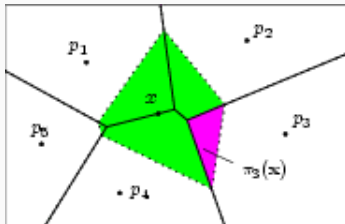
Coordonnées de Sibson

On se dote d'un ensemble de points d'entraînement $(\mathbf{p}_i)_{\{1 \dots I\}}$ dont on calcule le diagramme de Voronoï.

Coordonnées de Sibson

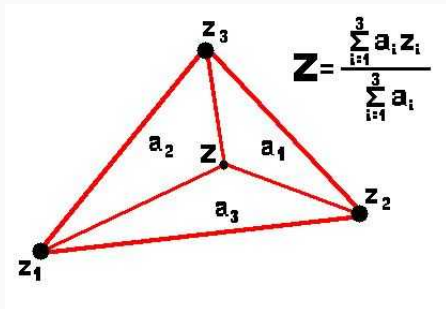
$$\mathbf{x} = \sum_{i=1}^I \lambda_i(\mathbf{x}) \mathbf{p}_i \quad \text{où} \quad \lambda_i(\mathbf{x}) = \frac{\pi_i(\mathbf{x})}{\pi(\mathbf{x})}$$

Illustration



Remarques et propriétés

Intepréétation des coordonnées de Sibson quand $l = 3$



Relations

$$\sum_{i=1}^l \lambda_i(x) = 1$$

Implémentation numérique

En C++, nous avons implémenté en 2D un programme qui interpole selon différents protocoles la couleur d'une image :

- Points d'entraînement choisis aléatoirement
- A erreur d'interpolation imposée, interpolation par ajout aléatoire de points
- A erreur d'interpolation imposée, interpolation par ajout hiérarchique de points
- A nombre de points fixés, minimisation du nombre de points d'interpolation nécessaires (optimisation).

Interpolation avec points d'entraînement choisis aléatoirement



Figure 1 – 2500 points d'entraînement



Figure 2 – 10000 points d'entraînement

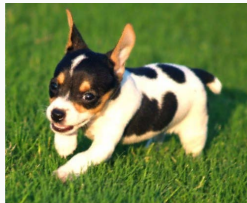


Figure 3 – 100000 points d'entraînement

Algorithm 1 Algorithme de calcul d'interpolée de Sibson

Require: Liste L des points d'entraînement initialisée aléatoirement

```
1: for  $x = 1 : w$  do  
2:   for  $y = 1 : h$  do  
3:     Calcul de l'interpolée de Sibson  $I_s(x, y)$  à partir de  $L$   
4:   end for  
5: end for
```

La gestion de l'erreur

$$MAE = \frac{\sum_{i=1}^n |\hat{x}_i - x_i|}{n}$$

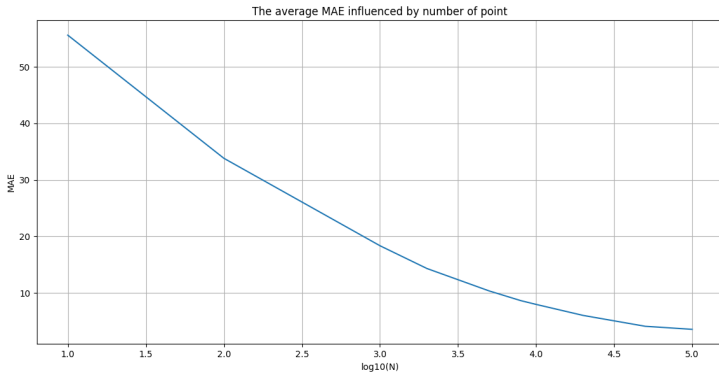


Figure 4 – MAE en fonction du nombre de points d'entraînement

Interpolation par ajout aléatoire de points

On travaille désormais en prenant en compte l'erreur d'interpolation. On va ajouter des points à notre ensemble d'entraînement jusqu'à satisfaire une condition d'inégalité sur l'erreur d'interpolation.

Algorithm 2 Algorithme de gestion d'erreur itératif

Require: Liste L des points d'entraînement initialisée (4 points au coin),
intensité I de l'image étudiée

```
1: erreur =  $\infty$ 
2: while erreur > seuil do
3:   ajout un point à  $L$  aléatoirement
4:   for  $x, y$  do
5:     Calcul de l'interpolée de Sibson  $I_s(x, y)$  à partir de  $L$ 
6:   end for
7:   erreur( $x, y$ ) =  $|I_s(x, y) - I(x, y)|$ 
8:   erreur = MOYENNE(erreur( $x, y$ ))
9: end while
```

La vitesse de l'interpolation itératif

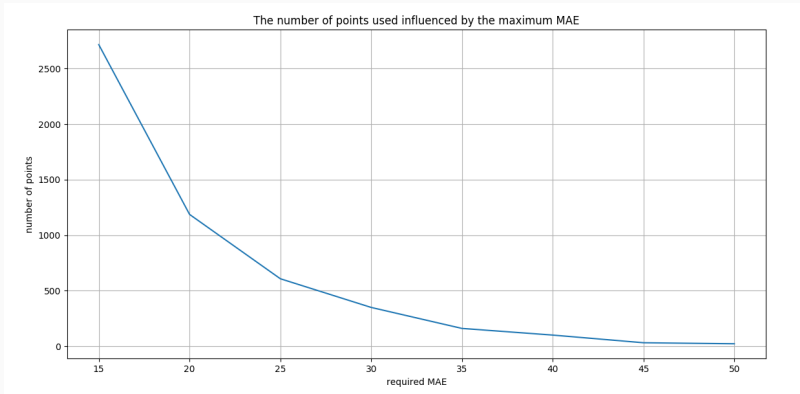


Figure 5 – MAE en fonction du nombre de points d'entraînement

Observation : Quand on choisit un petit seuil, le temps de calcul explose.

Interpolation par ajout hiérarchique de points

Pour accélérer la vitesse, on va ajouter les points hiérarchique. C'est-à-dire chaque point d'entraînement produisent un nouvel point dans chaque itération.

Algorithm 3 Algorithme de gestion d'erreur hiérarchique

Require: Liste L des points d'entraînement initialisée (4 points au coin), intensité I de l'image étudiée

```
1: erreur =  $\infty$ 
2: while erreur > erreur seuil do
3:   choix aléatoirement un point  $p$ 
4:   for  $l_i$  in  $L$  do
5:     Ajout du centre de segment  $pl_i$  à  $L$ 
6:   end for
7:   for  $x, y$  do
8:     Calcul de l'interpolée de Sibson  $I_s(x, y)$  à partir de  $L$ 
9:   end for
10:  erreur( $x, y$ ) =  $|I_s(x, y) - I(x, y)|$ 
11:  erreur = MOYENNE(erreur( $x, y$ ))
12: end while
```

Optimisation du nombre de points d'interpolation nécessaires

Maintenant, on propose une méthode qui optimiserait les positions des points d'interpolation pour minimiser l'erreur totale d'interpolation.

Algorithm 4 Algorithme de gestion d'erreur nécessaires

Require: Liste L des n points d'entraînement initialisée, intensité I de l'image étudiée

```
1: while True do
2:   for  $x, y$  do
3:     Calcul de l'interpolée de Sibson  $I_s(x, y)$  à partir de  $L$ 
4:     Calcul et sauvegarde de l'erreur en  $(x, y)$ 
5:   end for
6:   Le tri de l'erreur par l'ordre décroissante
7:   for  $i = 1 : \text{int}(\frac{n}{100}) - 4$  do
8:     Le remplacement du premier point sur la liste par le point dont l'erreur
       est le plus grand
9:   end for
10:  erreur = MOYENNE( $|I_s(x, y) - I(x, y)|$ )
11: end while
```

La condition de l'arrêt

Suivant l'idée de "early stop", on utilise la stratégie comme suit :

Algorithm 5 Algorithme de la condition de l'arrêt

Require: Un compteur c , le meilleur résultat obtenu e^* et le résultat dans cette itération e

```
1: if  $e < e^*$  then  
2:    $e \rightarrow e^*$   
3:    $c = 0$   
4: else if  $c < 10$  then  
5:    $c + 1 \rightarrow c$   
6: else  
7:    $e^* \rightarrow e$   
8:   stop  
9:   L'affichage de courbe de l'erreur en python automatiquement  
10: end if
```

Optimisation du nombre de points d'interpolation nécessaires



Figure 6 – Image après interpolation aléatoire ($n=10000$)

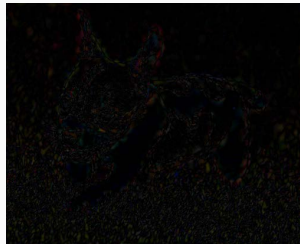


Figure 7 – Image après interpolation optimisée ($n=10000$)

Le processus de l'optimisation

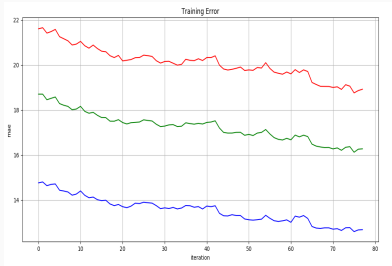


Figure 8 – Le courbe de l'erreur
($n=1000$)

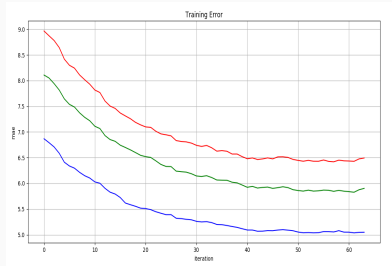





Figure 9 – Le courbe de l'erreur
($n=10000$)

Conclusion

-  R. Sibson, *A brief Description of Natural Neighbour Interpolation, Interpreting Multivariate Data* Woley, 1981
-  G. Farin, *Surfaces over Dirichlet Tessellations*, CAGD, 1990
-  *CGAL 2D and Surface Function Interpolation*

Phase de questions