Université de Bordeaux

Traitement d'image Avancée

Assignment 3

Graph Cut

Author:
Jimmy Gouraud

Supervisor:
Aurélie Bugeau

November 24, 2017

Contents

1 Introduction

Lors du premier rendu, nous avons implémenté une méthode de synthèse de texture ("Non-Parametric Sampling") qui consistait à créer une nouvelle texture en la synthétisant pixel par pixel. Les inconvénients de cette technique est que l'on doit traiter chaque pixel de l'image séparément ce qui entraîne un coup non négligeable en temps et créer du bruit dû en partie à la sélection aléatoire. Afin de corriger ces inconvénients, la technique d'Image Quilting [2] synthétise une texture par blocs et non par pixels.

2 Image Quilting

Pour créer une nouvelle texture avec la méthode d'Image Quilting :

1. On recopie en haut à gauche de notre image de destination un bloc prit aléatoirement dans la texture initiale.

2. On recherche un bloc qui "colle bien", c'est-à-dire dont la partie qui superpose le bloc précédent ressemble le plus possible à la partie qu'elle superpose.

3. On trouve une seam qui sépare au mieux les 2 blocs afin de sélectionner les pixels que l'on souhaite garder et ceux que l'on souhaite prendre du nouveau bloc.

Il faut itérer sur les parties 2. et 3. jusqu'à obtenir l'image de destination complètement remplie. Une illustration de la partie 3 est faite sur la Figure 1 à gauche.

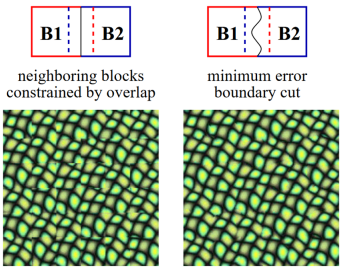

Figure 1: Superposition de blocs sans (à droite) et avec (à gauche) gestion de l'overlap

Dans le cadre du rendu, nous nous intéresserons uniquement à la partie 3, c'est-à-dire sur la façon de trouver une seam qui sépare au mieux les différents blocs. Nous verrons dans un premier temps l'implémentation à l'aide de la programmation dynamique et par la suite avec l'utilisation du Graph Cut.

3 Dynamic Programming

3.1 Algorithme

Pour calculer une seam séparant deux patch, nous utilisons une fonction de coût afin de pondérer chaque pixel de l'overlap. Ici, la fonction de coût correspond à la différence entre le pixel du patch A et le pixel du patch B correspond, auquel on ajoute la valeur minimale des voisins supérieur du pixel. On peut résumer la fonction de coût par :

$$E_{i,j} = SSD_{i,j} + min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

.

Algorithme 1 : Pseudo Code - Programmation dynamique

input :

- patch_size: taille du patch
- ov_size: taille de l'overlap
- patch_A: patch initialement collé
- patch_B: nouveau patch

output :

- cut: seam sur l'overlap

```
1 ov = compute_energy(patchA, patchB, patch_size, ov_size); // cost
    function
2 [min j] = find_min_last_rox(ov);
3 cut[patch_size] = j;
4 for i ← patch_size-1 : 1 do
5 |   [min j] = min(ov(i,j-1), ov(i,j), ov(i,j+1));
6 |   cut[i] = j;
7 end
```

3.2 Résultat

Les figures 3 et 2 ont été obtenu avec une taille de patch de 25 pixels et une taille d'overlap de 5 pixels.

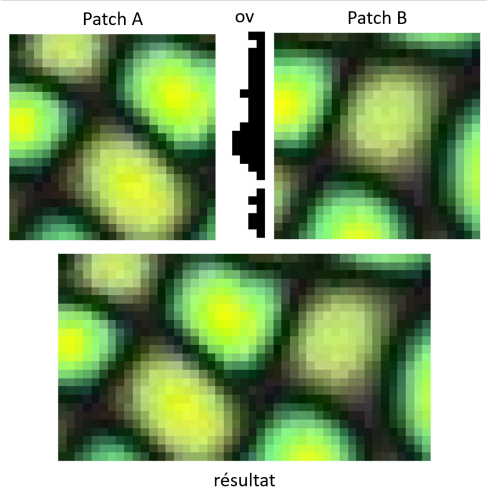

Figure 2: Fusion de 2 blocs avec un overlap de 5 pixels (programmation dynamique)

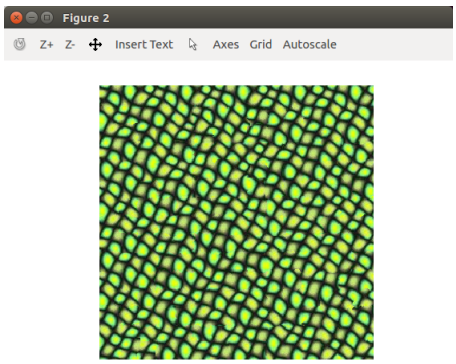

Figure 3: Résultat Image Quilting par programmation dynamique

4 Graph Cut

4.1 Algorithme

Le Graph Cut [1] est une technique de séparation des noeuds composant le graph en différentes parties appelées "classes". Ici, nous souhaitons l'utiliser pour trouver la seam qui sépare au mieux 2 patchs. Ainsi, nous aurons une classe 0 qui correspond au pixel déjà présents sur notre image de destination et une classe 1 qui correspond au pixel appartenant au nouveau bloc trouvé. Nous avons aussi besoin d'initier un "unary" qui est simplement le coût d'appartenir à l'une des deux classes. Lors de l'initialisation de cette unary, nous fixons des poids nulle à une classe et infinie à l'autre classe sur les pixels aux extrémités (à droite et à gauche). Les pixels entre ses deux bords ont autant de chance d'appartenir à l'une des deux classe qu'à l'autre, c'est pour cela que nous mettons des poids équivalents des deux côtés.
Ensuite, nous devons calculer le poids de chaque arête, cela est fait dans la matrice "pairwise". Le calcul de l'arête est simplement la somme des différence entre les pixels formant l'arête.
Avec toutes ses variables, il suffit de le donner à la fonction GCMex, qui nous retourne une matrice contenant l'appartenance des pixels à une classe ou l'autre : cela nous permet de séparer les pixels.

4.2 Résultat

Dans la figure 4, nous remarquons que le résultat est assez correctes. Néanmoins, dans la grandes majorités des cas, les résultats obtenus avec la méthode Graph Cut sont toujours moins bons (c'est-à-dire que l'on arrive à bien percevoir la séparation entre les deux patches) que les résultats obtenus avec la méthode de programmation dynamique.

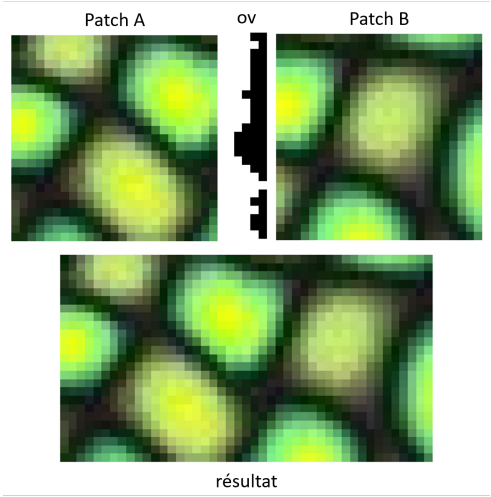

Figure 4: Fusion de 2 blocs avec un overlap de 5 pixels (programmation dynamique)

5 Conclusion

Pour faire de la synthèse de texture par Image Quilting, le mieux est d'utilisé la programmation dynamique que le Graph Cut. En effet, en plus d'être plus simple à mettre en place et plus facile à comprendre, elle est aussi plus rapide et les résultats obtenus sont meilleurs. Nous obtenons de meilleurs résultats avec la méthode par programmation dynamique qu'avec le graph Cut.

References

[1] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[2] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.