Projet synthèse de texture pour Algorithmique et Programmation Avancée

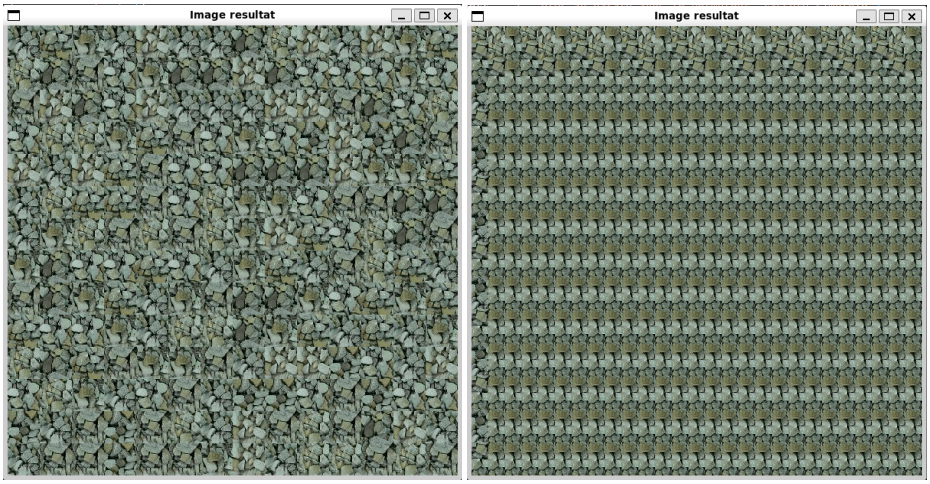
Auberval Florian, Behuet Timothée, Siaudeau Romain

December 13, 2022

L'objectif de ce TP est de déterminer la coupe optimale situé dans la zone de recouvrement de deux blocs. Pour résoudre ce problème, nous allons utiliser la programmation dynamique. Lien du dépot du projet: https://github.com/Mek-01/synthese_de_texture

Questions:

1 Expérimentez les algorithmes (a) et (b) fournis avec différentes images, et différents paramètres.



(a) algorithme a : bloc au hasard (0.05 s) (b) algorithme b (0.36 s)

On remarque que pour le cas de l'algorithme b, le meilleur bloc est lui-même, ce qui créer un motif non-désirable. Pour les algorithmes que nous allons implémenter nous utiliserons donc le permuteur.

2 Implémentez une variante de l'algorithme (a). Un générateur aléatoire de permutation de blocs permet d'équilibrer le nombre d'occurrences de chaque bloc.

Nous avons un indice max, qui représente les indices du tableau que nous pouvons sélectionner. On choisit une valeur aléatoire de 0 à indice max, et on échange de place la valeur choisie, avec celle de l'indice max. A chaque tour de boucle on décrémente indice max, pour réduire la sélection des indices possibles.

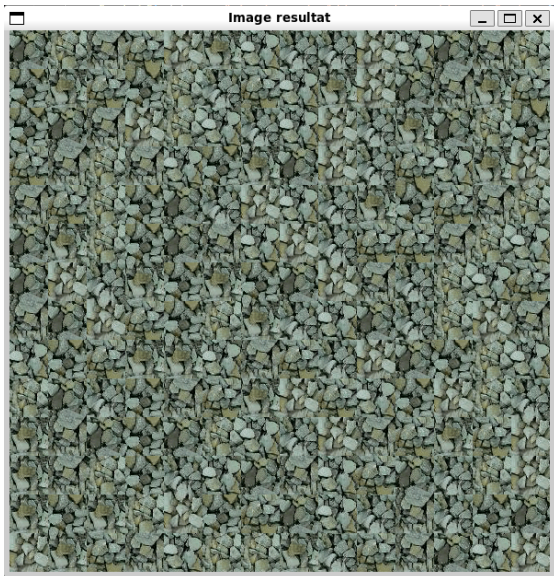


Figure 2: algorithme (a) variant (1.43 s)

cf "permuteur.cpp" pour voir le fonctionnement de la fonction de permutation

3 Question 3:

3.1 Que représente e[i, j]?

e[i, j] représente l'erreur entre les deux pixels de coordonnées i et j qui se chevauchent. Ils appartiennent aux bandes de recouvrement de deux blocs voisins. Elle est calculée à l'aide des écarts sur les canaux RGB des deux pixels qui se chevauchent.

3.2 Que représente E?

C'est le critère d'erreur entre deux bandes, qui est défini par la somme des erreurs de chaque pixels.

4 Quel est le rapport entre le coût de la coupe optimale et les E[i, j] ?

Le coût de la coupe optimale correspond à la somme de la norme des écarts RGB sur le chemin minimal. Donc E[i,j] est égal au coût de la coupe optimale de la ligne 0 jusqu'au pixel i, j.

5 Clarifiez la notion de coupe optimale sur l'exemple:

La coupe optimal est celle qui passe par le chemin dont le coût total est le plus petit. Par exemple, la coupe optimal pour l'exemple donnée serait:

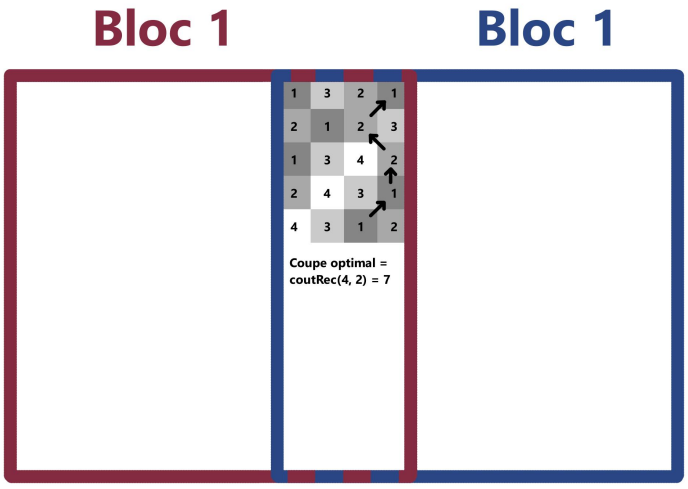


Figure 3: Exemple de coupe optimal à partir de la ligne i=4

Formule de récurrence (sans prendre en compte le dépassement de la bande): Avec e = matrice des erreurs (On suppose que les indices i et j commencent à 0).

$$coutRec\,(i,j) : \begin{cases} e\,[\,i,j\,] & si\ i = 0 \\ \min \begin{pmatrix} coutRec\,(i-1,j-1)\,, \\ coutRec\,(i-1,\ j\ )\,, \\ coutRec\,(i-1,j+1) \end{pmatrix} + e\,[\,i,j\,] & sinon \end{cases}$$

On a donc pour l'exemple donné:

$$coutRec(4,2) = 7$$

6 Estimez la complexité d'une solution récursive naïve pour le calcul de la coupe optimale.

Chaque chemin ouvre, dans le pire des cas, trois nouveaux chemins ce qui nous donne $3^{hauteur}$ chemins. De plus cela est fait autant de fois qu'il y a d'élément sur la première ligne, soit *largeur* fois. Ce qui nous donne:

$$complexité\ \ algorithmique = O(largeur * 3^{hauteur})$$

7 Implémentez une solution récursive sans calcul redondant.

Nous cherchons la coupe avec le coût minimal. Pour chaque pixels de la bande de recouvrement il est nécessaire de calculer la coupe de coût minimale.

Sachant qu'à chaque niveau s'ouvre trois (ou deux) nouveaux chemin. Il nous faut conserver celui possédant le coût le plus bas ainsi que la coupe qui lui est associée. Cependant certaine partie d'un chemin peuvent être commune pour plusieurs chemins. Pour éviter les calculs redondant nous stockons le coût minimal et la coupe associée. Nous nous dotons de 2 tableau tabCout et tabCoupe, puis nous appliquons la formule de récursion en faisant en sorte d'effectuer les calculs s'ils sont nécessaires, ou de récupérer les informations dans les tableaux sinon.

cf "raccordeur_recursif.cpp"

Figure 4: algo C récursif sans calcul redondant (0.65 s)

8 Implémentez une solution itérative.

Pour obtenir notre coupe optimale avec l'algorithme itératif, nous avons initialisé le tableau de coût par ceux de la première ligne, ainsi que le tableau des coupes optimals par des valeurs par défauts.

Pour les remplir nous allons de la ligne 1 jusqu'à hauteur-1, et on calcul pour chaque pixels sur l'axe x de la bande de recouvrement toutes les coupes possibles depuis cette ligne. Puis nous les stockons dans des tableaux à chaque étape y, le coût minimal et sa coupe associée pour le pixel x, y.

Puis nous parmi toutes les coupes calculées, nous cherchons le départ qui donne le coût minimal, afin d'obtenir le chemin de la coupe optimale.

cf "raccordeur_iteratif.cpp"

Figure 5: algo C iteratif (0.40 s)

9 Quelle est la complexité globale de l'algorithme (c) en prenant en compte tous les paramètres ?

Pour obtenir le coût minimal et donc la coupe associée, notre algorithme doit parcourir l'ensemble de la bande de recouvrement une seule fois. C'est pour cela que la complexité de notre algorithme itératif est en:

$$complexité\ algorithmique\ de\ l'itératif = O(largeur * hauteur)$$

En pratique, l'algorithme récursif sans redondance est 1.6 fois plus lent que l'algorithme itératif pour l'image gravier.tif. (en terme de temps d'execution on a 0.65 s pour le récursif contre 0.40 s pour l'itératif).