INFO2050-1 - Programmation Avancée

Projet 3 : Mise en page automatique d'une bande dessinée

Aliaksei Mazurchyk
Antoine Sadzot

20 décembre 2017

Université de Liège

1 Algorithme par programmation dynamique

1.1 Approche par recherche exhaustive

1.1.1 Comic

1.1.2 Seam carving

Pour créer une couture, on part d'un pixel de la première rangée. De là, on a le choix entre trois directions pour continuer la couture à la rangée suivante. Pour avancer d'encore une rangée, on a encore le choix entre trois directions... On a donc (en prenant en compte les bords gauche et droit) une complexité $O(m * 3^n)$ où m est le nombre de pixels de départ à la première rangée et n le nombre de pixels en hauteur.

1.2 Formulation récursive de la fonction de coût

1.2.1 Comic

1.2.2 Seam carving

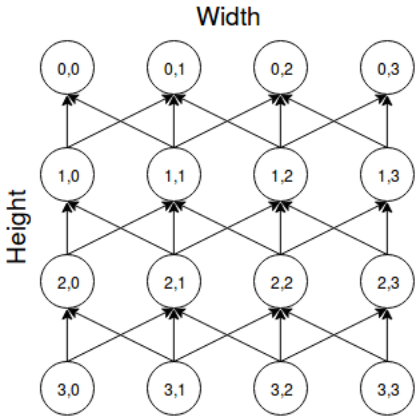
$$C(i,j) = \begin{cases} E(0,j) & si\ i == 0 \\ E(i,j) + min(C(i-1,j-1), C(i-1,j), C(i-1,j+1)) & sinon \end{cases}$$

1.3 Graphe des appels récursifs pour un problème de petite taille

1.3.1 Comic

1.3.2 Seam carving

Graphe des appels récursif



1.4 Algorithme des fonctions de coût

1.4.1 Comic

1.4.2 Seam carving

La fonction **E(image,i,j)** est la fonction qui calcule l'énergie d'un pixel. Le tableau **energies** contient les coutures d'énergie minimales pour atteindre chaque pixel depuis la première rangée. Le tableau **moves** contient le déplacement à faire (gauche, centre, droite) pour suivre une couture de bas en haut.

```
C (image, energies, moves)
for i=0 to n
        for j=0 to m
        {
        if i==0
                energies[i,j]=E(image,i,j)
        else
                if E(image,i-1,j-1) < E(image,i-1,j)
                        tmp=E(image,i-1,j-1)
                else
                        tmp=E(image,i-1,j)
                if tmp < E(image,i-1,j+1)
                        min=tmp
                else
                        min=E(image,i-1,j+1)
                energies[i,j]=E(image,i,j)+min
                if min == E(image,i-1,j-1)
                        moves[i,j]= -1
                else if min = E(image,i-1,j)
                        moves[i,j]=0
                else
                        moves[i,j]=1
        }
return energies, moves
```

1.5 Complexité en temps et en espace

1.5.1 Comic

1.5.2 Seam carving

La complexité en temps de l'algorithme est $\Theta(n * m)$ La complexité en espace est $\Theta(3 * n * m)$. Car il y a trois tableaux de dimension n*m. L'image en elle même, plus un tableau pour enregistrer les coutures d'énergie et un tableau pour enregistrer les déplacements à faire pour suivre une couture de bas en haut.

2 Fonctions de réduction et d'augmentation d'images

2.1 Choix d'implémentation

Les deux fonctions **ImageWidthReduction** et **ImageWidthexpansion** sont très proches dans leur fonctionnement. La seule différence est lorsqu'on crée une nouvelle image temporaire, respectivement plus grande ou plus petite de 1 pixel.

Une fonction **energie** renvoie l'énergie calculée d'un pixel donné en argument avec l'image auquel il appartient.

La fonction **seams** utilise la fonction **energie** pour remplir le tableau contenant les coutures d'énergie minimales pour chaque pixel de l'image transmise en argument. Elle remplit également le tableau indiquant les déplacements à effectuer pour suivre les coutures minimales depuis la dernière ligne de l'image.

La fonction **selectedSeam** parcourt le tableau des coutures pour enregistrer le numéro de colonne de chaque pixel de la couture minimale dans un vecteur de taille correspondant à la hauteur de l'image.

Les opérations décrites ci-dessous sont effectuées un nombre de fois correspondant au nombre de pixels que l'on souhaite ajouter ou diminuer sur la largeur de l'image.

Au sein des fonctions **ImageWidthReduction** et **ImageWidthexpansion**, pour chaque iteration, on sélectionne la couture d'énergie minimale au moyen des fonctions décrites ci-dessus. Ensuite, pour générer une nouvelle image de largeur +/- 1 à chaque iteration, on travaille avec deux pointeurs vers deux images temporaires. L'image pointée par "new" est initialement de la même taille que l'image originelle. L'image pointée par "tmp" est l'image augmentée ou réduite d'un pixel en largeur par rapport à new. Lorsqu'une couture est sélectionnée sur l'image de "new", l'image de "tmp" est construite en tenant compte de la couture. Ensuite, l'image pointée par "new" est détruite, et "new" pointe vers l'image de "tmp". A l'itération suivante, "tmp" pointe vers une nouvelle image, plus petite ou plus grande que "new".

2.2 Complexité en temps et en espace

Les complexités en temps et en espace sont identiques pour les deux fonctions.

2.2.1 Complexité en espace

En plus de l'image originelle, deux autres images avec un écart de largeur égal à 1 sont générées pour les calculs. En fonction des itérations, leur largeur évolue entre la largeur originelle et la largeur cible. A ces deux images s'ajoutent le tableau contenant les coutures et le tableau contenant les déplacements. Les largeurs des deux tableaux évoluent comme les images. Et à cela s'ajoute un vecteur de taille "n", contenant les numéros de colonne des pixels impliqués dans la couture sélectionnée. La complexité en espace est donc maximale au début de la fonction : $\theta(3 * (n * m) + n * (m - 1) + n) = \theta(4 * n * m)$

2.2.2 Complexité en temps

Pour diminuer ou augmenter une image de 1px en largeur la complexité en temps est la suivante : L'image est parcourue une fois pour déterminer les coutures($\theta(n * m)$). Elle est ensuite parcourue en largeur et en hauteur pour déterminer la couture d'énergie minimale ($\theta(n + m)$). L'image est ensuite parcourue une nouvelle fois pour générer une nouvelle image. Là la complexité change en fonction de si l'on veut agrandir ou rétrécir l'image :

- Agrandir : une colonne en plus est générée ($\theta(n * m + n)$). On a donc au total une complexité $\theta(2n * m + m + 2n)$
- Rétrécir : une colonne en moins est parcourue ($\theta(n * m - n)$). On a donc au total une complexité $\theta(2n * m + m)$

Il faut maintenant prendre en compte le nombre k d'itérations à effectuer pour modifier l'image de k pixels en largeur. En faisant attention que la taille des tableaux diminue à chaque itération

Rétrécissement d'une image :

$$T_{n,m,k} = \sum_{i=0}^{k} 2n(m-i) + m - i$$

$$= (2n+1)\sum_{i=0}^{k}(m-i)$$

$$= (2n+1)(\sum_{i=0}^{k} m - \sum_{i=0}^{k} i)$$

$$= (2n+1)(k * m - \frac{k(k+1)}{2})$$

$$= k(2n+1)(m - \frac{(k+1)}{2})$$

Lorsque k est négligeable par rapport à m, la complexité est proche de $\theta(2knm)$

Agrandissement d'une image :

En s'aidant de la formule précédente, on arrive facilement à la formule suivante :

$$T_{n,m,k} = (2n+1)\sum_{i=0}^{k}(m-i) + \sum_{i=0}^{k} 2n$$

$$= k2n + k(2n+1)(m - \frac{(k+1)}{2})$$

Encore une fois, lorsque k est négligeable par rapport à m, et en prenant en compte que $k2n$ est négligeable par rapport au reste de la formule, on a une complexité proche de $\theta(2knm)$