Agrandir des images avec des splines cubiques

Dans ce TP, vous mettrez en place un outil permettant de réaliser une interpolation d'un ensemble de points et appliquerez cela à un algorithme d'agrandissement d'images

1 Introduction à l'interpolation par spline cubique (juste lire, rien à coder)

L'interpolation polynomiale est une méthode qui a ses limites. L'une d'entre elles est l'oscillation forte qui peut apparaître entre les nœuds d'interpolation quand ces derniers sont nombreux, comme vous avez pu le voir lors du dernier TP.

Une autre méthode, l'interpolation par morceaux, existe afin de proposer une méthode aux résultats plus stables. L'idée est de ne réaliser une interpolation que sur un sous ensemble de points consécutifs. Par exemple, si on possède 12 points, on pourrait réaliser des interpolations de paquets de 4 points consécutifs, et "recoller" les polynômes ainsi obtenus.

Les splines cubiques font partie des méthodes d'interpolation par morceaux, et consistent à faire passer, entre chaque couple de points contigus (x_i, y_i) et (x_{i+1}, y_{i+1}), un polynôme d'interpolation de degré 3 (d'où le nom cubique), appelé P. Une méthode d'interpolation par spline cubique, appelée *méthode de Catmull Rom*, propose le principe suivant pour calculer le polynôme d'interpolation entre chaque paire de points contigus :

$$\begin{cases} P(x_i) = y_i \\ P(x_{i+1}) = y_{i+1} \\ \\ P'(x_i) = \dfrac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}} \\ \\ P'(x_{i+1}) = \dfrac{y_{i+2} - y_i}{x_{i+2} - x_i} \end{cases} \tag{1}$$

On rajoute donc, en plus d'une contrainte de position, une contrainte sur la dérivée du polynôme aux points x_i et x_{i+1}, afin que les différents polynômes ainsi calculés se "recollent" de façon plus naturelle.

Si on possède cinq points $((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5))$, on calculera donc un polynôme d'interpolation de degré 3 (une spline cubique) entre (x_1, y_1) et (x_2, y_2), un polynôme d'interpolation de degré 3 (une spline cubique) entre (x_2, y_2) et (x_3, y_3), un polynôme d'interpolation de degré 3 (une spline cubique) entre (x_3, y_3) et (x_4, y_4) et un polynôme d'interpolation de degré 3 (une spline cubique) entre (x_4, y_4) et (x_5, y_5).

Si l'on nomme P la spline d'interpolation passant entre (x_i, y_i) et (x_{i+1}, y_{i+1}), et G la spline d'interpolation passant entre (x_{i+1}, y_{i+1}) et (x_{i+2}, y_{i+2}), on remarque que l'on a :

$$\begin{cases} P(x_{i+1}) = G(x_{i+1}) \\ P'(x_{i+1}) = G'(x_{i+1}) \end{cases}$$

Ainsi, les splines des différentes parties se recollent correctement, en ayant la même pente. Pour le premier point (x_0, y_0) et pour le dernier point (x_n, y_n) de l'ensemble de points, on aura :

$$\begin{cases} P'(x_0) = \dfrac{y_1 - y_0}{x_1 - x_0} \\ \\ P'(x_n) = \dfrac{y_n - y_{n-1}}{x_n - x_{n-1}} \end{cases}$$

2 Calculer les coefficients de la spline cubique (rien à coder, mais il faudra faire un peu de calcul)

Cherchons à calculer les coefficient de la spline cubique $P(x)$ passant entre les points $(x_i; y_i)$ et $(x_{i+1}; y_{i+1})$. Le polynôme P étant de degré 3, il peut s'écrire

$$P(x) = ax^3 + bx^2 + cx + d$$

Ce polynôme est, comme dit précédemment, tel que

$$\begin{cases} P(x_i) = y_i \\ P(x_{i+1}) = y_{i+1} \\ P'(x_i) = k_1 \\ P'(x_{i+1}) = k_2 \end{cases} \tag{2}$$

avec, comme dit précédemment, $k_1 = \dfrac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}$ et $k_2 = \dfrac{y_{i+2} - y_i}{x_{i+2} - x_i}$ (mais, pour des raison de simplifications de nos équations, on conservera k_1 et k_2).

Résoudre directement cet ensemble d'équations n'est pas impossible, mais ce sera long. Un simple changement de variable permet de simplifier énormément les calculs, aussi nous allons procéder de cette façon.

Nous posons $t = \frac{x - x_i}{x_{i+1} - x_i}$. On va chercher un polynôme $S(t) = et^3 + ft^2 + gt + h$ tel que $P(x) = S(t)$. On remarque que si $x = x_i$ alors $t = 0$, et si $x = x_{i+1}$ alors $t = 1$. On aura donc

$$\begin{cases} P(x_i) = S(0) = y_i \\ P(x_{i+1}) = S(1) = y_{i+1} \\ P'(x_i) = S'(0) = k_1 \\ P'(x_{i+1}) = S'(1) = k_2 \end{cases} \tag{3}$$

Questions

1. A vous de calculer, sur papier, les valeurs de e, f, g et h du polynôme $S(t)$. A partir de l'ensemble de quatre équations donné juste avant, utilisez tout d'abord la première équation pour trouver h et la troisième question pour trouver g. Vous pourrez ensuite combiner la seconde et quatrième équation pour trouver e et f.

2. Il faut maintenant trouver les valeurs de a, b, c et d du polynôme $P(x)$. Pour cela, vous savez que $P(x) = S(t) = S(\frac{x - x_i}{x_{i+1} - x_i})$. Développez l'expression du polynôme $S(\frac{x - x_i}{x_{i+1} - x_i})$, et identifiez les coefficient de x^3, x^2, x et les coefficients constants, et vous aurez ainsi trouvé a, b, c et d.

3 Implémentation en C d'une fonction d'interpolation par spline cubique

Maintenant que les calculs ont été réalisés, il faut passer à l'implémentation en C d'une fonction d'interpolation par spline cubique.

Questions

1. Vous devez tout d'abord réfléchir à la structure qui contiendra la spline cubique interpolant, par morceaux, un ensemble de points $(x - 1, y_1), (x_2, y_2), ..., (x_n, y_n)$. Dans cette structure que vous appellerez **maspline**, vous devrez pouvoir stocker le nombre de points n qui sont interpolés, les valeurs de leurs abscisses $x_1, x_2, ...x_n$ (les ordonnées ne sont pas nécessaires), et pour chaque polynôme de degré 3 interpolant une pair de points (il y en a combien ?), vous devrez stocker les valeurs des coefficients a, b, c et d. Proposez une structure permettant de stocker toutes ces informations.

2. Proposez une fonction **void calcul_une_spline**(x_i, y_i, x_{i+1}, y_{i+1}, k_1, k_2, &a, &b, &c,

&d), qui prendra en entrée deux paires de nœuds d'interpolation, ainsi que les valeurs k_1 et k_2 que devrait avoir la dérivée de la spline aux extrémités, et remplit a, b, c et d avec les valeurs des coefficients du polynôme de degré 3 respectant les contraintes de spline.

3. Proposez une fonction **maspline calcul_spline(**tab_x**,** tab_y**,** n**)**, qui prendra en entrée un tableau de n abscisses et un tableau de n ordonnées représentant les nœuds d'interpolation, et renverra une structure **maspline** remplie avec les valeurs des splines cubiques interpolant les points.

4. Réalisez l'interpolation par spline des points ci-dessous, qui sont les mêmes que ceux donnés dans le précédent TP, et pour lesquels l'interpolation polynomiale n'avait pas donné de bons résultats. Les résultats obtenus sont-ils plus satisfaisants d'un point de vue esthétique ?

x	2	4	5	6.5	8	10	15	20	25	30
y	3.5	0.5	7	8	-3	6	-2	-1	3	0

4 Application de l'interpolation par spline pour agrandir une image

Nous allons tenter d'agrandir une image de façon assez simple : entre chaque colonne et chaque ligne de l'image, nous rajouterons deux nouvelles colonnes et deux nouvelles lignes. Imaginons par exemple une image de 4 pixels par 4 pixels comme ci-dessous :

23	46	230	120
3	31	12	89
10	19	67	101
4	187	203	198

Dans cette (petite) image, si nous rajoutons deux nouvelles lignes et deux nouvelles colonnes entre chaque ligne et chaque colonne de l'image, nous obtiendrons une image de 10 pixels par 10 pixels comme ci-dessous :

23			46			230			120
3			31			12			89
10			19			67			101
4			187			203			198

Cette image possède de nombreux pixels vides qu'il faut remplir. Nous commencerons tout d'abord par nous occuper de chaque ligne de l'image possédant quelques pixels vides : il s'agit, si l'on numérote les lignes en commençant par 0, des lignes 0, 3, 6 et 9 sur notre exemple. Sur chaque ligne, nous réaliserons une interpolation par spline cubique afin de trouver les valeurs manquantes.

Par exemple, sur la ligne 0, nous avons la valeur 23 à la colonne 0, la valeur 46 à la colonne 3, la valeur 230 à la colonne 6 et la valeur 120 à la colonne 9. Nous réaliserons donc une interpolation par spline avec, comme nœuds d'interpolation, les points $(0, 23), (3, 46), (6, 230), (9, 120)$. Cette interpolation nous permettra d'obtenir les valeurs à mettre dans les colonnes 1, 2, 4, 5, 7, et 8.

Une fois toutes les lignes à un indice multiple de 3 remplies, nous effectuerons une interpolation par spline colonne par colonne, sur toutes les colonnes. Une fois cette dernière étape réalisée, nous aurons une valeur dans chacun des pixels de notre nouvelle image.

Questions

Réalisez un programme **zoom**, qu'il faudra rendre en fin de séance (suivez les instructions de rendu de la section suivantes), qui prendra en entrée une image png et génèrera en sortie une image de taille différente, selon la méthode décrite précédemment.

5 Rendu de votre programme

Vous devez rendre le travail réalisé pour le TP9 sur Moodle.
Ce rendu doit suivre un format spécifique. Votre programme devra s'exécuter à l'aide de la commande

```
./zoom entree.png z sortie.png
```

— **entree.png** doit être l'image qui sera lue et agrandie en entrée.

— **z** sera un réel positif représentant le coefficient d'agrandissement ou rétrécissement de l'image. La nouvelle image sera donc de hauteur égale à z fois la hauteur de l'image, et de même pour la largeur.

— **sortie.png** sera l'image de sortie, résultat de l'agrandissement ou du rétrécissement de l'image d'entrée.

Votre programme devra être compilé à l'aide de la commande

```
gcc zoom.c lodepng.c -lm -o zoom
```

Si votre ligne de compilation est plus complexe, vous devrez alors la spécifier dans un *Makefile*. Tous vos fichiers devront être directement placés dans un fichier zip (et ne pas être dans un sous dossier du fichier zip), dont le nom sera

```
VotreNom_VotrePrenom_VotreNumeroEtudiant.zip
```

sans aucun espace (si votre nom ou votre prénom contiennent un espace, ne les faîtes pas figurer). Il n'est pas utile de joindre les fichiers *lodepng.c* et *lodepng.h* à votre rendu. Les formats de compression acceptés sont .zip, .tgz, .tar.gz, .7z, .rar.

Si vous avez un binôme, il vous faudra écrire son nom, prénom et numéro d'étudiant séparés par des espaces, en plus des vôtres, dans un fichier *binome.txt*, dont le format sera

```
Nom1 Prenom1 Numero_etudiant1
Nom2 Prenom2 Numero_etudiant2
```

Vous ne devez rendre le fichier qu'une seule fois par binôme.