Licence de Sciences et Technologies — Université de Perpignan Via Domitia
S3 - mention Mathématiques — V. Zucca

Projet d'Algorithmique et de Programmation

Troisième Partie : morphing

DATE LIMITE DE RENDU : 22 / 12 / 2023

1 Préliminaires : un peu d'algèbre linéaire

Afin de pouvoir mener le projet à bien, nous allons avoir besoin de quelques routines d'algèbre linéaire, en particulier déterminer une transformation affine à partir d'un système d'équations.

Une transformation affine est une transformation géométrique qui préserve l'alignement des points : c'est à dire que l'image d'une droite est une droite (rotations, homothétie, translation, etc...). C'est la composition d'une transformation linéaire et d'une translation.

Une transformation affine du plan peut être représentée par une matrice 2×3 soit 6 coefficients.

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{pmatrix} \in \mathbb{R}^6$$

L'image d'un point $P(x, y)$ du plan par une transformation affine, représentée par une matrice A, se calcule par le produit :

$$A \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{R}^2$$

1.1 Résolution d'un système linéaire via le pivot de Gauss

Afin de résoudre un système d'équations de façon numérique, l'approche standard est de triangulariser le système à l'aide du pivot de Gauss puis de résoudre le système triangulaire.

Créer une fonction `gauss(A : list[list[float]], y : list[float]) -> None` qui va transformer le système $\texttt{A} \cdot \texttt{x} = \texttt{y}$ en un système triangulaire supérieur équivalent à l'aide du pivot de Gauss.

Créer également une fonction `solve_upper_triangular(A : list[list[float]], y : list[float]) -> list[float]` qui va renvoyer la solution `x` du système $\texttt{A} \cdot \texttt{x} = \texttt{y}$ où `A` est une matrice triangulaire supérieure.

Vous pourrez vérifier vos fonctions en multipliant la matrice `A` de départ par le résultat trouvé et vérifier que vous trouverez (à peu près) le vecteur `y` de départ.

1.2 Détermination d'une transformation affine

En connaissant un couple $P(x, y)$, $Q(x', y')$ tel que Q soit l'image de P par une transformation affine inconnue on obtient un système de deux équations à six inconnues :

$$\begin{cases} x' &= a_{0,0} \cdot x + a_{0,1} \cdot y + a_{0,2} \\ y' &= a_{1,0} \cdot x + a_{1,1} \cdot y + a_{1,2} \end{cases}$$

Pour pouvoir déterminer une transformation affine inconnue A il faut connaitre trois couples de points $P_0, P_1, P_2, Q_0, Q_1, Q_2$ tels que Q_i soit l'image de P_i par A. En l'écrivant sous forme matricielle on obtient :

$$\begin{pmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{0,0} \\ a_{0,1} \\ a_{0,2} \\ a_{1,0} \\ a_{1,1} \\ a_{1,2} \end{pmatrix}$$

Créer une fonction `get_affine_transform(start : list[list[int]], end : list[list[int]]) -> list[list[float]]` prenant en paramètre deux listes de trois points. `start` est une liste contenant les 3 points de départ P_0, P_1, P_2 et `end` est une liste contenant les trois images Q_0, Q_1, Q_2 des points contenus dans `start` par la même transformation affine. La fonction renverra la matrice de la transformation affine.

2 Morphing

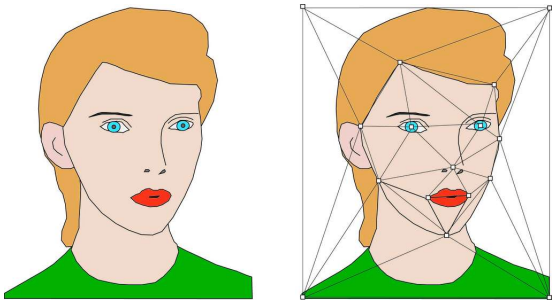
2.1 Principe

Le morphing d'image est un des effets spéciaux applicables à des images photographiques ou cinématographiques. À partir de deux images de départ, il permet de fabriquer une troisième image qui sera la position intermédiaire des deux premières.

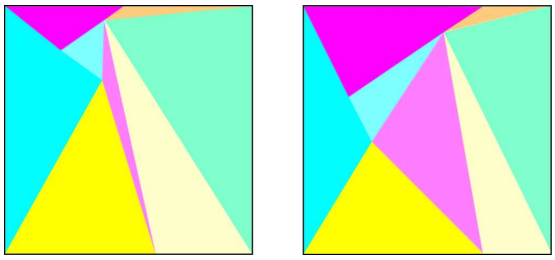
Cela permet de réaliser des vieillissements accélérés dans les films, d'éviter de dessiner plusieurs dessins intermédiaires dans les animés. Il a été utilisé au cinéma dans plusieurs films (Indiana Jones et la dernière croisade, Terminator 2, Titanic, ...). Son principe est simple :

1. Les deux dessins doivent être découpés en deux systèmes de triangles compatibles (même nombre, même disposition).
2. Pour chaque paire de triangles on peut déterminer les coordonnées d'un triangle intermédiaire.
3. On peut donc déterminer la transformation permettant de passer du triangle de départ au triangle intermédiaire et du triangle intermédiaire au triangle d'arrivé.
4. Une fois les transformations déterminées, on peut calculer la couleur de chaque pixel du triangle intermédiaire.

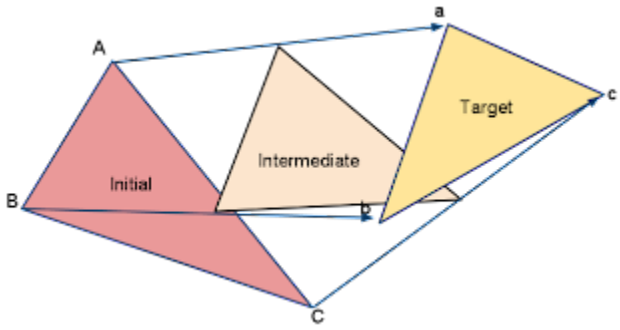
Les sommets des triangles doivent correspondre à des éléments importants du dessin
— les coins de l'image
— pour un visage : menton, oreilles, oeil, front



Une fois les deux dessins découpés, on obtient deux ensembles de triangles : chaque triangle sur l'image de départ correspondant à un triangle sur l'image d'arrivée.



Une fois que nous avons les coordonnées des sommets de chaque paire de triangles nous pouvons déterminer les coordonnées d'un triangle intermédiaire.



On rappelle que étant donné deux points $P(x_P, y_P)$ et $Q(x_Q, y_Q)$ les points du segments $[PQ]$ ont pour coordonnées :

$$\begin{cases} x &= (1-\alpha)x_P + \alpha x_Q \\ y &= (1-\alpha)y_P + \alpha y_Q \end{cases} \quad \text{avec } \alpha \in [0,1]$$

Étant donné que l'on dispose des coordonnées des sommets du triangle de départ et du triangle intermédiaire on peut déterminer la transformation affine pour passer de l'un à l'autre.

De la même manière on peut déterminer la transformation permettant de passer du triangle intermédiaire au triangle final.

Pour chaque pixel du triangle intermédiaire on peut donc :

— Chercher la couleur du pixel correspondant sur le triangle de départ C_d

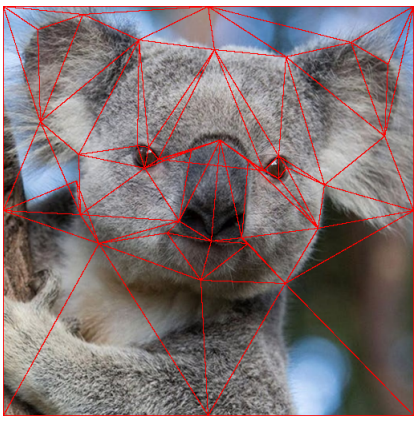
— Chercher la couleur du pixel correspondant sur le triangle d'arrivé C_a

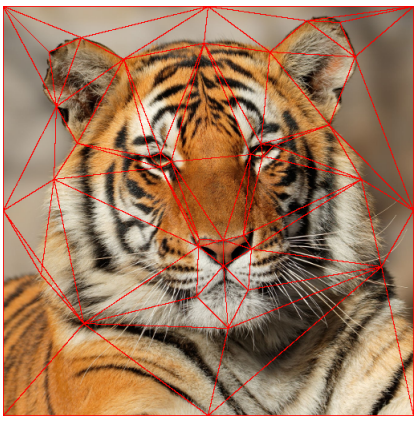
— Obtenir la couleur du pixel en calculant : $(1-\alpha)C_d + \alpha C_a$

On applique ce procédé à chaque paire de triangles des images de départ et finale pour obtenir l'image intermédiaire.

2.2 Récupération de la triangulation de nos images

Le fichier `points.txt` contient les coordonnées des points correspondant aux sommets des triangles sur nos deux images.





Le fichier est enregistré sous le format suivant :

```
$ head points.txt
# Koala (colonne x ligne) Tigre (colonne x ligne)
41
167 174 183 188
176 202 196 201
193 191 209 194
181 176 196 183
324 205 308 185
343 215 323 187
353 201 337 174
343 187 321 172
```

La première ligne, commençant par un #, est une ligne de commentaires. Le nombre situé sur la seconde ligne, ici 41, correspond au nombre de points selectionnés sur les deux images.

Les lignes suivantes indiquent les coordonnées des points $(P_i)_i$ sur l'image du koala et $(Q_i)_i$ correspondant sur l'image du tigre.

Les deux premiers nombres de chaque ligne correspondent respectivement aux numéros de colonne et de ligne du point P_i. De même, les deux derniers nombres correspondent respectivement aux numéros de colonne et de ligne du point Q_i correspondant à P_i sur l'image du tigre. Par exemple, les premiers points inscrits dans le fichier ont pour coordonnées :

$$P_0(174, 167) \qquad Q_0(188, 183)$$

Une fois que nous aurons récupéré nos listes de points il faudra savoir comment ils sont reliés. La fichier `triangles.txt` contient la liste des triangles formés par les points contenus dans nos listes. Le fichier a le format suivant :

```
$ head triangles.txt
72
17 29 38
30 37 34
40 30 34
39 31 36
31 40 36
26 30 40
31 26 40
14 18 33
18 38 33
```

L'entier sur la première ligne correspond au nombre de triangles utilisés pour trianguler l'ensemble de l'image. Les lignes suivantes correspondent aux indices des sommets de chaque triangle dans nos listes de points.

Par exemple, le premier triangle indiqué dans le fichier a pour sommets P_{17}, P_{29} et P_{38} sur la première image et son équivalent a pour sommet Q_{17}, Q_{29} et Q_{38} sur la seconde image.

2.3 Remplissage des triangles de l'image intermédiaire

On suppose que nous disposons à présent de la liste de nos triangles sur nos deux images. Pour chaque paire de triangles $P_{i_0}P_{i_1}P_{i_2}$ et $Q_{i_0}Q_{i_1}Q_{i_2}$ nous allons calculer les sommets du triangle intermédiare $R_{i_0}R_{i_1}R_{i_2}$ pour un paramètre α.

Nous allons ensuite déterminer la couleur des pixels des points à l'intérieur du triangle intermédiaire. Pour cela, nous allons devoir déterminer si un point est situé à l'intérieur ou à l'extérieur du triangle.

Soit $M(x_M, y_M)$, $P_1(x_1, y_1)$ et $P_2(x_2, y_2)$ trois points du plan. On s'intéresse à la quantité suivante :

$$\det(M, P_1, P_2) = (x_M - x_2) \cdot (y_1 - y_2) - (x_1 - x_2) \cdot (y_M - y_2)$$

En notant $d_1 = \det(M, P_1, P_2)$, $d_2 = \det(M, P_2, P_3)$ et $d_3 = \det(M, P_3, P_1)$, un point M sera situé à l'intérieur du triangle de sommmets P_1, P_2, P_3 lorsque les nombres d_1, d_2 et d_3 seront tous trois de même signe. Dans le cas contraire M sera situé à l'extérieur du triangle.

Pour déterminer la couleur d'un pixel situé au point P dans un triangle de l'image intermédiaire de paramètre α on procède de la façon suivante :

- on détermine la tranformation affine A_d permettant de passer du triangle intermédiaire au triangle de départ.
- on détermine la tranformation affine A_f permettant de passer du triangle intermédiaire au triangle final.
- on détermine la couleur C_d du pixel $A_d(P)$ sur l'image de départ
- on détermine la couleur C_f du pixel $A_f(P)$ sur l'image de finale
- on obtient la couleur du pixel situé au point P en calculant $(1 - \alpha)C_d + \alpha C_a$.

Attention : les résultats de $A_d(P)$ et $A_f(P)$ ne seront pas forcément des vecteurs à coordonnées entières. Il faudra arrondir les coordonnées des vecteurs trouvés afin d'avoir les coordonnées des pixels correspondant.

3 Travail à rendre

Vous rendrez un fichier `morphing.py` (ou `morphing.ipynb` si vous préférez les notebooks) qui permettra d'exécuter un programme permettant de calculer les 27 images intermédiaires de paramètre $\alpha = 0, 0.04, 0.08, 0.12, \ldots, 0.92, 0.96, 1$ des deux images `koala.ppm` et `tigre.ppm`. On considèrera que `koala.ppm` est l'image de départ.

Les images seront nommées respectivement `morphing_0.ppm`, `morphing_4.ppm`, `morphing_8.ppm`, ..., `morphing_96.ppm` et `morphing_100.ppm`.

Vous rendrez également un bref rapport (3 pages maximum), soit au format `pdf`, soit directement dans le notebook. Dans ce rapport vous justifierez vos choix de développements, de vos structures de données, des paramètres de vos fonctions et vous détaillerez le fonctionnement de votre programme.

morphing_0.ppm

morphing_20.ppm

morphing_40.ppm

morphing_60.ppm

morphing_80.ppm

morphing_100.ppm