

TP7-2013. Morphing suivant l'algorithme de Beier-Neely

EXERCICE 1 (Correspondance de deux segments orientés). On considère deux images I0 et I1 toutes deux de tailles [N1,N2] et

- deux pixels P_0, Q_0 de I0 ($P_0=[j_0 \ k_0]$, $Q_0=[j_0 \ k_0]$);
- deux pixels P_1, Q_1 de I1 ($P_1=[j_1 \ k_1]$, $Q_1=[j_1 \ k_1]$).

Ces deux paires de pixels induisent donc des segments orientés $[P_0, Q_0]$ et $[P_1, Q_1]$ respectivement sur chacune des deux images I0 et I1, ces deux segments orientés étant supposés en correspondance (ainsi préparés pour le **morphing**).

(1) Construire une routine

[PosWarp0, PosWarp1, Mu, Nu]

= BeierNelly1aux (I0,I1,[j0 k0 jj0 kk0],[j1 k1 jj1 kk1],s);

qui, pour tout pixel $x = (j, k)$ d'une image PosWrap (pour PositionPixelsWrap) au départ constituée de zéros) de même taille [N1,N2] que I :

- calcule le vecteur unitaire

$$\begin{aligned} \vec{t}(s) &= \frac{(1-s)(Q_0-P_0) + s(Q_1-P_1)}{\|(1-s)(Q_0-P_0) + s(Q_1-P_1)\|} \\ &= \frac{(1-s)(j_0-j_0, k_0-k_0) + s(j_1-j_1, k_1-k_1)}{\|(1-s)(j_0-j_0, k_0-k_0) + s(j_1-j_1, k_1-k_1)\|} ; \end{aligned}$$

et son image $\vec{n}(s)$ par rotation de $\pi/2$ (dans le sens direct);

- calcule les coordonnées $\mu_s(x)$ et $\nu_s(x)$ du vecteur

$$xx = x - \left((1-s)(j_0, k_0) + s(j_1, k_1) \right) = x - ((1-s)P_0 + sQ_0)$$

dans le repère orthogonal direct d'origine le point

$$(1-s)(j_0, k_0) + s(j_1, k_1)$$

et de vecteurs directeurs les vecteurs

$$(1-s)(j_0-j_0, k_0-k_0) + s(j_1-j_1, k_1-k_1) = (1-s)(Q_0-P_0) + s(Q_1-P_1)$$

et $\vec{n}(s)$.

- renvoie comme PosWarp0(x) la valeur de la position de pixel réassigné (pour l'image I0) :

$$(j_0, k_0) + \mu_s(x)(j_0-j_0, k_0-k_0) + \nu_s(x)\vec{n}(0)$$

et pour PosWarp1(x) la valeur de la position de pixel réassigné (pour l'image I1) :

$$(j_1, k_1) + \mu_s(x)(j_1-j_1, k_1-k_1) + \nu_s(x)\vec{n}(1)$$

(les tableaux `PosWarp0` et `PosWarp1` seront donc déclarés ici comme des tableaux de taille `[N1,N2,2]`).

- renvoie aussi les matrices `Mu` et `Nu` correspondant aux coordonnées $(\mu_s(\mathbf{x}), \nu_s(\mathbf{x}))$, \mathbf{x} désignant le pixel (j, k) de la matrice.

- (2) Réaliser à partir de la routine `BeierNeelyaux` (exploitée comme code auxiliaire) une routine

```
>> [Warp0, Warp1, Warp, Mu, Nu]
    = BeierNeely (I0, I1, [j0 k0 jj0 kk0], [j1 k1 jj1 kk1], s);
```

réassigne les images `I0` et `I1` sur les images s -« intermédiaires » `Warp0` et `Warp1` (tenant compte de la correspondance du segment joignant les pixels (j_0, k_0) et (j_{j_0}, k_{k_0}) dans `I0` avec le segment joignant les pixels (j_1, k_1) et (j_{j_1}, k_{k_1}) dans `I1`) lorsque s (dans $[0, 1]$) figure le paramètre de déformation, puis interpole ces deux images `Warp0` et `Warp1` en

$$\text{Warp} = (1 - s) \text{Warp0} + s \text{Warp1}.$$

Ce code placera dans un premier temps les deux images au centre d'un cadre de taille $(N1, N2)$ où $N1 = m + 2*MM$, $N2 = n + 2*NN$, avec

$$MM = \text{ceil}((\sqrt{m^2 + n^2} - m)/2), \quad NN = \text{ceil}((\sqrt{m^2 + n^2} - n)/2),$$

ce aux fins de rendre possibles les rotations nécessaires sans perte de données.

EXERCICE 2 (correspondance de K segments orientés et moyennisation de Beier-Neely). On considère une liste de K segments en correspondance

$$[P_{k,0}, Q_{k,0}] \longleftrightarrow [P_{k,1}, Q_{k,1}]$$

sur deux images `I0` et `I1` de même taille $[m, n]$, codés par deux matrices `J0` et `J1` toutes les deux de taille $(K, 4)$: la ligne d'indice k de `J0` correspond à la position des pixels $P_{k,0}$ et $Q_{k,0}$ de l'image `I0` tandis que la ligne d'indice k de `J1` correspond à la position des pixels des points en correspondance $P_{k,1}$ et $Q_{k,1}$ de l'image `I1`.

- (1) Modifier le code `BeierNeely` conçu à l'exercice 1 (question 2) en un code

```
>> [Iinit0, Iinit1, Warp] =
    MultiBeierNeely(I0, I1, J0, J1, a, b, p, s);
```

qui, étant donnés trois paramètres $a \in]0, 1]$, $b \in [1/2, 2]$, $p \in [0, 1]$ et $s \in [0, 1]$:

- calcule, pour chaque valeur $k=1:\text{size}(J0, 1)$, la norme du vecteur

$$V_{k,s} := (1 - s)(Q_{k,0} - P_{k,0}) + s(Q_{k,1} - P_{k,1}).$$

- calcule, toujours pour chaque valeur de k , la matrice `Dk(s, :, :)` des distances $Dk(\mathbf{s}, \mathbf{x})$ des pixels $\mathbf{x} = (j, k)$ d'une matrice de taille $[N1=m+2*MM, N2=n+2*NN]$ (voir à l'exercice précédent, question 2, pour la définition de `MM` et `NN`) au segment joignant les deux pixels $(1 - s)P_{k,0} + sQ_{k,0} + (MM, NN)$ et $(1 - s)P_{k,1} + sQ_{k,1} + (MM, NN)$; on notera que cette distance vaut :

$$Dk(\mathbf{s}, \mathbf{x}) = \begin{cases} \| \mathbf{x} - (1 - s)Q_{k,0} - sP_{k,0} - (MM, NN) \| & \text{si } \text{Muk}(\mathbf{x}) > 1 \\ |\text{Nu}_k(\mathbf{x})| & \text{si } 0 \leq \text{Muk}(\mathbf{x}) \leq 1 \\ \| \mathbf{x} - (1 - s)Q_{k,0} - sP_{k,0} - (MM, NN) \| & \text{si } \text{Muk}(\mathbf{x}) < 0 \end{cases}$$

où les matrices Muk et Nuk ont été calculées *via* `BeierNeely.m` à partir de deux images $I0$ et $I1$ de taille (m, n) et des données $J0_k$ et $J1_k$ correspondant à la k -ième paire de segments en correspondance, donc via

```
>> [Warpk0, Warpk1, Warpk, Muk, Nuk] =
    BeierNeely(I0, I1, J0(k, :), J1(k, :), s);
```

– calcule, pour chaque valeur de k , la matrice des coefficients de pondération :

```
    alphak(s, :, :) := (||Vk,s|| ./ (a + Dk(s, :, :))). ^ b
```

– calcule enfin la moyenne pondérée des divers « Warp » $Warpk$ obtenus comme précédemment grâce à la routine

```
>> [Warpk0, Warpk1, Warpk, Muk, Nuk] =
    BeierNeely(I0, I1, J0(k, :), J1(k, :), s);
```

à partir des deux images $I0$ et $I1$, chacun pour la k -ième paire de segments correspondants $([P_{k,0}, Q_{k,0}] \rightarrow [P_{k,1}, Q_{k,1}])$:

$$\frac{\sum_{k=1}^K \text{alphak}(s, :, :) * \text{Warpk}}{\sum_{k=1}^K \text{alphak}(s, :, :)}$$

– renvoie également en sortie les images encadrées `Iinit0` et `Iinit1` de taille $[N1, N2]$ conditionnées pour le morphing.

On réalise ainsi le « morphing » entre $I0$ et $I1$ avec K correspondances de segments orientés prescrites.

- (2) Tester le code en prenant pour $I0$ et $I1$ respectivement les images `lena` et `einstein`. Les correspondances (respectivement) des segments correspondant à l'œil gauche, à l'œil droit, à la bouche et aux nez, sont données par les deux matrices :

$$J0 = \begin{bmatrix} 266 & 251 & 274 & 288 \\ 272 & 318 & 268 & 351 \\ 351 & 269 & 349 & 323 \\ 273 & 314 & 326 & 316 \end{bmatrix}$$

$$J1 = \begin{bmatrix} 211 & 198 & 225 & 235 \\ 223 & 282 & 213 & 317 \\ 310 & 220 & 312 & 307 \\ 205 & 248 & 279 & 252 \end{bmatrix}$$

(les positions des extrémités des segments ont été repérées sur les figures `image(I0)` et `image(I1)` grâce au curseur de `MATLAB`). Modifier pour les tests les valeurs des paramètres a, b, p .