Morphing

RYAN BOUCHOU, X
X, X
X

CURSUS INGÉNIEUR
PREMIÈRE ANNÉE
GÉNIE INFORMATIQUE

20 janvier 2024

Tuteur entreprise :
ÉLISABETH X
prénom.nom@entreprise.com

Résumé

Table des matières

1 Cas des polygones simples **3**
1.1 Généralités 3
1.2 Morphing naif 3
1.2.1 Principe et notation 3
1.2.2 Cas des images matricielles 4
1.3 Méthode L-A 7
1.3.1 Principe et notation 7

Annexes **10**

Références **11**

1 Cas des polygones simples

En premier lieu, nous allons nous intéresser au cas des polygones simples. Ce faisant, ceci sera l'occasion pour nous de faire notre premier pas dans l'*informatique graphique*, en traitant des cas élémentaires du problème de la morphose.

1.1 Généralités

Conventions Pour toute la suite, et sauf mention contraire, on se place dans le $\mathbb{R}$-evn $\mathbb{R}^2$ muni de la norme euclidienne. On notera pour tout vecteur $x \in \mathbb{R}^2$, $X \in \mathcal{M}_{2,1}(\mathbb{R})$ la matrice colonne associée à x dans la base canonique de $\mathbb{R}^2$.

Définition 1.1. Soit n un entier naturel non nul. On appelle *polygone simple* de $\mathbb{R}^2$ tout n_uplet $P = (p_1, \dots, p_n)$ de $\mathbb{R}^2$ tels que :

1. Les segments ne se croisent pas, c'est-à-dire que pour chaque paire de segments $[p_i, p_{i+1}]$ et $[p_j, p_{j+1}]$ (où p_{n+i} est p_i), les segments ne partagent pas de points autres que les sommets.
2. Chaque sommet p_i est partagé par exactement deux segments.

On notera p_{i^+} le segment $[p_i, p_{i+1}]$ et p_{i^-} le segment $[p_i, p_{i-1}]$. Enfin, on notera $\mathbb{P}$ l'ensemble des polygones simples de $\mathbb{R}^2$.

Définition 1.2. Soit $P \in \mathbb{P}$ un polygone simple de $\mathbb{R}^2$.

1. On appelle *ordre* de P le nombre n de composantes de P.
2. On appelle *arête* de P tout segment p_{i^+} ou p_{i^-} pour $i \in \{1, \dots, n\}$.
3. On appelle *sommet* de P tout vecteur p_i pour $i \in \{1, \dots, n\}$.

Subséquemment, pour n un entier naturel non nul, on note $\mathbb{P}_n$ l'ensemble des polygones simples de $\mathbb{R}^2$ d'ordre n.

Définition 1.3. Soit $P \in \mathbb{P}$ un polygone simple de $\mathbb{R}^2$. On appelle *intérieur* de P, noté $\overset{\circ}{P}$, l'ensemble des points $x \in \mathbb{R}^2$ tels que x est à gauche de chaque arête de P.

Situation À ce stade, l'enjeu de cette première partie est de déterminer un algorithme permettant la morphose d'un polygone simple P vers un autre polygone simple Q. Pour ce faire, il nous faut nous intéresser aux conditions d'une telle transformations, ainsi qu'à sa réalisation.

1.2 Un premier algorithme de morphing

1.2.1 Principe et notation

Principe L'idée de cet algorithme est de déformer progressivement le polygone P en un autre polygone Q. Pour ce faire, une première approche consiste à déformer chaque sommet p_i de P en un sommet q_i de Q par une interpolation linéaire. Ainsi, on obtient une suite de polygones $(P_k)_{0 \leq k \leq N}$ où N est le nombre d'images intermédiaires voulues et tels que $P_0 = P$, $P_N = Q$.

> Pour que l'algorithme soit correct, il est nécessaire que les polygones P et Q soient de même ordre.

Notation Soit $n, N > 0$ et $(P_k)_{0 \leq k \leq N}$ une suite de polygones de $(\mathbb{P}_n)^{\mathbb{N}}$. On notera $p_1^{(k)}, \ldots, p_n^{(k)}$ les sommets de P_k.

Ce faisant, pour le calcul des images intermédiaires on donne l'algorithme suivant :

Données : $P, Q \in \mathbb{P}_n$ deux polygones, $N > 0$ le nombre de frames
Résultat : Une suite de polygones (P_k)
$P^* \leftarrow (P_1, \ldots, P_N)$
pour $k \leftarrow 0$ **à** N **faire**
  $t \leftarrow \frac{k}{N}$
  $P_k = (p_1^{(k)}, \ldots, p_n^{(k)})$
  pour $i \leftarrow 1$ **à** n **faire**
    $p_i^{(k)} \leftarrow (1-t) * p_i + t * q_i$
  fin
fin
retourner P^*

Algorithme 0 : générationFramesNaif

1.2.2 Cas des images matricielles

Principe Dans le cadre de l'exercice, la donnée du problème est consituée de deux images matricielles P et Q de taille $L \times l$ représentant respectivement des polygones simples de couleur même couleur. Naturellement, plusieurs méthodes algorithmiques peuvent être utilisées pour encoder l'information géométrique portée par une image. Entre autre, des algorithme de *détection de contours* ou de *segmentation* peuvent être utilisés. Toutefois, et par soucis de simplicité, nous allons considérer que l'information géométrique est directement encodée par l'utilisateur lors de la saisie des *points de contrôle*. De plus, pour chaque pixel de coordonnées (i, j), on a :

$$\begin{cases} \mathbb{I}_P(i,j) = 1 & \text{si } (i,j) \in \overset{\circ}{P} \\ \mathbb{I}_P(i,j) = 0 & \text{sinon} \end{cases}$$

> Pour que l'algorithme soit correct, il est nécessaire que les images P et Q soient de même taille. De plus, si les points de contrôle saisis par l'utilisateur ne correspondent pas aux sommets des polygones, les résultats de l'algorithme peuvent être inattendus.

Propriété 1.1. Soit $u, v \in \mathbb{R}^2$ deux vecteurs. Alors u est à gauche de v si et seulement si $det(u, v) > 0$.

DÉMONSTRATION : Admis. o.ε.δ.

Propriété 1.2. Soit $p = [u, v], (u, v) \in \mathbb{R}^2$ un segment et $x \in \mathbb{R}^2$ un vecteur. Alors x est à gauche de p si et seulement si $det(p, [u, x]) > 0$.

Démonstration : Soit $u, v \in \mathbb{R}^2$ deux vecteurs. On note θ l'angle orienté entre u et v. Alors,

$$\begin{aligned} u \text{ est à gauche de } v &\iff \det(u,v) > 0 && (1)\\ &\iff \|u\| \cdot \|v\| \sin(\theta) > 0 && (2)\\ &\iff \sin(\theta) > 0 && (3)\\ &\iff \theta \in \,]0, \pi[&& (4)\\ &\iff u \text{ est dans le demi-espace à gauche de } v && (5)\end{aligned}$$

o.ε.δ.

Ce faisant, pour le calcul des images intermédiaires, et la determination de l'intérieur du polygone, de on donne la suivante :

Données : Une liste de points *tab* représentant le polygone, un point P
Résultat : Un booléen indiquant si le point P est à l'intérieur du polygone

```
nbp ← taille de tab pour i ← 0 à nbp − 1 faire
    A ← tab[i] B ← tab[(i + 1) mod nbp]
    D ← (B.x − A.x, B.y − A.y) T ← (P.x − A.x, P.y − A.y)
    d ← det(D, T)
    si d < 0 alors
        retourner Faux
    fin
fin
retourner Vrai ;
```

Algorithme 1 : isInside

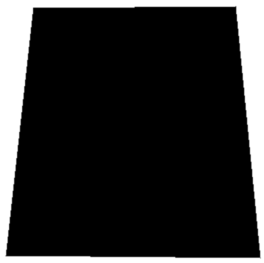
On en déduit l'algorithme naif pour le mophing de formes unies simples :

Données : Deux images matricielles P et Q de taille $L \times l$, un entier N
Résultat : Une suite d'images matricielles P^*

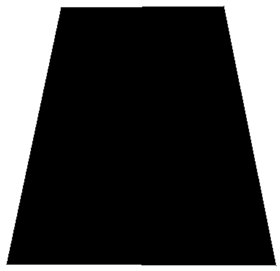
```
nbp ← taille de tab
P* ← générationFramesNaif(P, Q)
pour i ← 0 à nbp − 1 faire
    pour x ← 0 à L − 1 faire
        pour y ← 0 à l − 1 faire
            if(isInside(P*_i, (x, y))) P*_i[x][y] ← color
        fin
    fin
fin
retourner P*
```

Algorithme 2 : morphingNaif

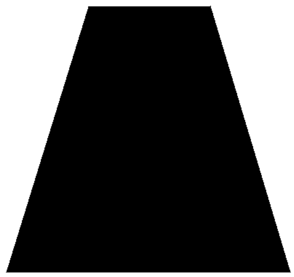
Résultats Considérons une exécution de l'algorithmes sur deux instances du problème.



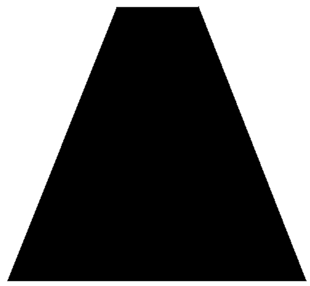
(a) Image initiale



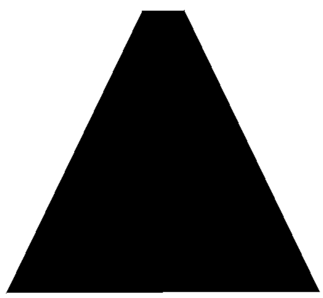
(b) Image intermédiaire 1



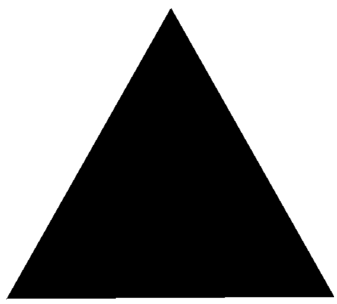
(c) Image intermédiaire 2



(d) Image intermédiaire 3



(e) Image intermédiaire 4



(f) Image finale

FIGURE 2 – Résultats de l'algorithme de morphing

> Dans le cas de polygones simples, relativements semblables dans leur géométrie, l'algorithme de morphing naif donne des résultats satisfaisants.

En pratique Toutefois, il apparaît qu'une telle approche ne préserve pas les propriétés géométriques du polygone. En effet, les images intermédiaires peuvent contenir des segments qui se croisent, ou des sommets qui ne sont pas partagés par exactement deux segments. Dans un tel cas, la transformations ne paraît pas naturelle.

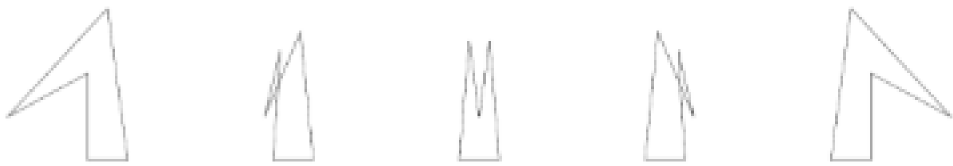


Figure 3 – Interpolation linéaire, observez les auto-intersections. [1]

Ainsi, il est nécessaire de trouver une approche plus rigoureuse pour le morphing de polygones simples.

1.3 Morphing par interpolation longeur-angle

1.3.1 Principe et notation

Principe L'idée de cette méthode, issue des travaux de Sederberg [2], est de déformer progressivement le polygone P en un autre polygone Q, en interpolant linéairement la mesure des angles entre chaque arêtes consécutives, et leurs normes respectives. Ce faisant, l'avantage est de préserver les propriétés géométriques souhaitées pour un rendu visuel naturel. On parle d'*invariance par mouvement rigide.* de morphing améliorée est.

Notation Soit $n, N > 0$ et $(P_k)_{0 \leq k \leq N}$ une suite de polygones de $(\mathbb{P}_n)^{\mathbb{N}}$. On pose, pour $k \in \{0, \ldots, N\}$:

→ $l_i^{(k)}$ la longueur de l'arête $p_{i^+}^{(k)}$,

→ $\theta_i^{(k)}$ l'angle entre les arêtes $p_{i^-}^{(k)}$ et $p_{i^+}^{(k)}$.

Subséquemment, pour une image intermédiaire $k \in \{0, \ldots, N\}$, avec $t = \frac{k}{N}$ on a :

$$
\begin{aligned}
l_i^{(k)} &= (1-t)l_i + tq_i \\
\theta_i^{(k)} &= (1-t)\theta_i + t\theta_i
\end{aligned}
$$

Liste des Algorithmes

0 générationFramesNaif 4
1 isInside 5
2 morphingNaif 5

Table des figures

2 Résultats de l'algorithme de morphing 6
3 Interpolation linéaire, observez les auto-intersections. [1] 7

Annexes

Références

[1] Mélanie Cornillac. *Morphing multirésolution de courbes.* Modélisation et simulation, Université de Grenoble, 2010. NNT : ff, tel-00581474f.

[2] T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2-d shape blending : An intrinsic solution to the vertex path problem. In *Computer Graphics (SIGGRAPH 93 Proceedings)*, volume 27, pages 15–18, 1993.