Université de Lorraine — UFR MIM

Master 1 Informatique — Module Initiation à la Recherche

Morphing

Implémentation de l'algorithme de Beier & Neely avec détection automatique de landmarks faciaux et variante exponentielle

réalisé par : Es-sebbar Karam

Encadrant : Dominique Michel

Langage : C++17, OpenCV 4, dlib

Référence : Beier & Neely, SIGGRAPH 1992

Année académique : 2025–2026

Résumé

Ce rapport présente l'étude et l'implémentation de l'algorithme de morphing d'images proposé par Beier et Neely en 1992 dans le cadre de la conférence SIGGRAPH. Le morphing d'images consiste à produire une séquence d'images interpolant de façon continue et visuellement cohérente entre deux images sources, en combinant une déformation géométrique (*warp*) et un fondu colorimétrique (*cross-dissolve*).

L'algorithme repose sur le concept de *segments directeurs* (feature lines) : des paires de segments définies manuellement ou automatiquement sur chaque image, guidant la déformation locale par une pondération spatiale. Nous détaillons les fondements mathématiques de la transformation — coordonnées locales (u, v), reconstruction du pixel source, fonction de poids — ainsi que la stratégie de *reverse mapping* garantissant l'absence d'artefacts.

Contribution du projet. Par rapport à la formulation originale de Beier et Neely, ce travail apporte plusieurs contributions distinctes. Premièrement, nous proposons et analysons une variante de la fonction de poids à décroissance exponentielle, en comparant analytiquement et visuellement son comportement à la formulation originale. Deuxièmement, nous intégrons la bibliothèque dlib afin d'automatiser la détection des landmarks faciaux et le placement des segments, supprimant ainsi la phase manuelle de définition des correspondances. Enfin, nous proposons une piste d'amélioration de la méthode de Beier et Neely par l'intégration de l'algorithme de triangulation de Delaunay afin d'obtenir des transformations plus robustes et mieux adaptées à certaines déformations géométriques.

Mots-clés : morphing d'images, field morphing, segments directeurs, reverse mapping, Beier-Neely, OpenCV, dlib, landmarks faciaux, déformation géométrique, fonction de poids exponentielle.

Table des matières

1 Introduction

1.1 Contexte et Motivation

Le morphing d'images est une technique fondamentale de traitement de l'image et d'animation numérique, consistant à produire une transition visuellement continue entre deux images. Contrairement au simple fondu (*cross-dissolve*), le morphing intègre une déformation géométrique qui aligne les structures visuelles des deux images avant de les fondre. Le résultat est une séquence d'images intermédiaires dans laquelle les formes se transforment progressivement et naturellement, sans les artefacts de double exposition caractéristiques du fondu brut.

Depuis ses premières apparitions dans l'industrie cinématographique au début des années 1990 — notamment dans le clip *Black or White* de Michael Jackson (1991), puis dans les productions *Terminator 2* et *Forrest Gump* — le morphing est devenu un outil standard des effets visuels et de l'animation. Ses applications s'étendent aujourd'hui à l'imagerie médicale (interpolation entre coupes anatomiques), à la réalité augmentée (filtres faciaux temps réel) et à l'art numérique.

L'algorithme de Beier et Neely [1], présenté à SIGGRAPH 1992, constitue une avancée majeure par rapport aux méthodes antérieures fondées sur des grilles de contrôle [2]. En remplaçant les points ou grilles rigides par des *segments directeurs* à champ d'influence local, il offre un contrôle à la fois intuitif et expressif.

1.2 Contribution du projet

Ce rapport ne se limite pas à reproduire l'algorithme de l'article de référence. Notre contribution se situe à deux niveaux complémentaires.

Variante exponentielle de la fonction de poids. La fonction de poids originale de Beier et Neely présente une décroissance polynomiale en $1/d^b$, qui laisse une influence résiduelle non nulle à grande distance. Nous proposons une variante à décroissance exponentielle $w_i^{\exp} = \text{length}_i^p \cdot e^{-k \cdot \text{dist}_i}$, que nous comparons analytiquement et expérimentalement à la formule originale. Cette variante réduit le nombre de paramètres de contrôle et offre une coupure plus nette de l'influence locale.

Automatisation par détection de landmarks. L'algorithme original requiert un placement manuel des segments de correspondance entre les deux images. Nous intégrons la bibliothèque dlib [4] et le modèle de prédiction de forme de Kazemi et Sullivan [5] pour automatiser cette étape sur des images faciales : les 68 landmarks détectés sont convertis en 67 paires de segments consécutifs sans aucune intervention humaine.

Perspective d'amélioration par triangulation de Delaunay. Enfin, nous proposons une amélioration potentielle de la méthode de Beier et Neely par l'intégration de la triangulation de Delaunay. Cette approche permettrait de mieux structurer les déformations géométriques locales et d'obtenir des transformations plus stables et plus cohérentes sur certaines régions complexes du visage.

1.3 Organisation du Rapport

Le rapport est structuré comme suit. Le chapitre 2 établit les bases théoriques du morphing et justifie le choix du *reverse mapping*. Le chapitre 3 détaille la transformation induite par une paire de segments unique. Le chapitre 4 généralise au cas multi-segments et analyse la fonction de poids, y compris notre variante exponentielle. Le chapitre 5 présente les pseudo-codes formels de tous les algorithmes. Le chapitre 6 décrit l'implémentation C++ et l'intégration de dlib. Le chapitre 7 analyse comparativement les paramètres et discute les résultats. Enfin, le chapitre 8 présente la triangulation de Delaunay comme perspective d'amélioration.

2 Fondements du Morphing d'Images

2.1 Définition Formelle

Définition 2.1 (Morphing d'images)**.** *Soient I_0 et I_1 deux images définies sur le même domaine discret $\Omega \subset \mathbb{Z}^2$. On appelle* morphing *toute famille d'images $\{F(t)\}_{t\in[0,1]}$ telle que $F(0) = I_0$, $F(1) = I_1$, et la transition $t \mapsto F(t)$ soit visuellement continue.*

La formule générale du morphing de Beier-Neely est :

$$F(t) = (1-t)\cdot \mathrm{warp}(I_0,\, A \to M(t)) \;+\; t\cdot \mathrm{warp}(I_1,\, B \to M(t)) \tag{2.1}$$

où A et B désignent respectivement les ensembles de segments directeurs définis sur I_0 et I_1, et $M(t)$ est l'ensemble des segments intermédiaires obtenus par interpolation linéaire à l'instant t.

2.2 Interpolation des Segments Directeurs

Pour chaque paire de segments (A_i, B_i), le segment intermédiaire $M_i(t)$ est obtenu par interpolation linéaire de ses extrémités :

$$M_i(t) = \big((1-t)\,P_i^A + t\,P_i^B,\; (1-t)\,Q_i^A + t\,Q_i^B\big) \tag{2.2}$$

Cette interpolation garantit que $M_i(0) = A_i$ et $M_i(1) = B_i$, assurant la cohérence aux bornes du morphing : à $t = 0$, le warp est identité sur I_0, et à $t = 1$, il est identité sur I_1.

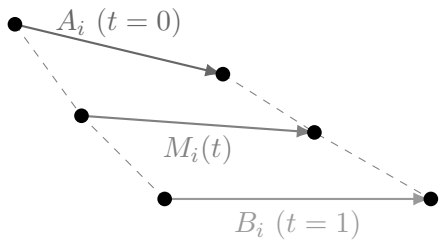


FIGURE 2.1 – Interpolation linéaire des segments directeurs de A_i vers B_i via $M_i(t)$.

2.3 Reverse Mapping : Justification Mathématique

La stratégie de calcul du warp peut être abordée de deux façons opposées.

2.3.1 Forward Mapping et ses Limites

En *forward mapping*, on parcourt chaque pixel X' de l'image source et on calcule sa position $f(X')$ dans l'image destination. Cette approche souffre de deux défauts fondamentaux liés aux propriétés de la fonction f :

Propriété 2.1. *Le forward mapping n'est en général ni injectif ni surjectif. Il en résulte :*

- *des* collisions *: plusieurs pixels sources projetés sur le même pixel destination (violation de l'injectivité),*
- *des* trous *: des pixels de destination sans antécédent dans la source (violation de la surjectivité).*

2.3.2 Reverse Mapping

En *reverse mapping*, on parcourt chaque pixel X de l'image *destination* et on calcule son antécédent $X' = f^{-1}(X)$ dans la source, puis on lit la couleur en X'.

Propriété 2.2 (Surjectivité du reverse mapping)**.** *Par construction, le reverse mapping garantit que tout pixel de l'image destination est rempli exactement une fois. L'image résultante ne présente donc aucun trou.*

De plus, puisque X' n'est pas en général à des coordonnées entières, l'interpolation bilinéaire est appliquée sur les quatre pixels voisins de X' dans l'image source, ce qui améliore la qualité visuelle en évitant les artefacts d'aliassage.

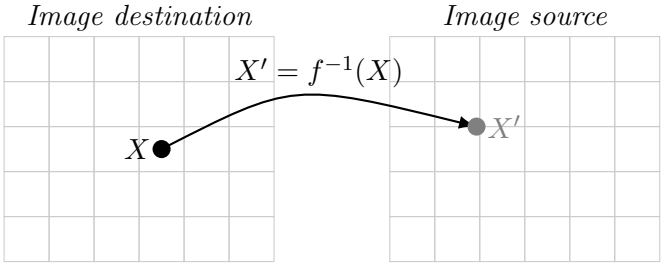


FIGURE 2.2 – Principe du reverse mapping : pour chaque pixel X de la destination, on calcule $X' = f^{-1}(X)$ dans la source.

3 Transformation par une Paire de Segments

3.1 Coordonnées Locales (u, v)

L'idée centrale de Beier et Neely est d'associer à chaque pixel X un système de coordonnées local défini relativement à un segment directeur PQ. Ce système capture la position du pixel à la fois le long du segment (coordonnée u) et perpendiculairement à celui-ci (coordonnée v).

Définition 3.1 (Coordonnée u — position longitudinale)**.**

$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2} \tag{3.1}$$

$u = 0$ correspond à l'extrémité P, $u = 1$ à l'extrémité Q, et $u \in]0, 1[$ à un pixel situé dans la projection perpendiculaire du segment.

Définition 3.2 (Coordonnée v — distance perpendiculaire signée)**.**

$$v = \frac{(X - P) \cdot \perp (Q - P)}{\|Q - P\|} \tag{3.2}$$

où $\perp (Q - P) = (-(Q_y - P_y),\, Q_x - P_x)$ désigne le vecteur perpendiculaire à $Q - P$ de même norme. Le signe de v indique de quel côté du segment se trouve X.

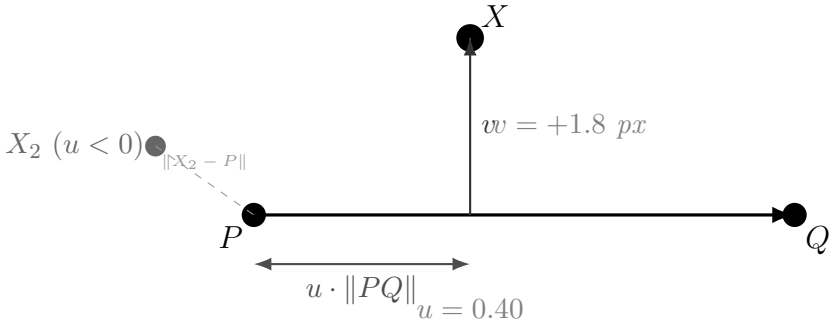


FIGURE 3.1 – Coordonnées locales (u, v) d'un pixel X par rapport au segment PQ. Pour $u < 0$, la distance utilisée est $\|X - P\|$.

3.2 Reconstruction du Pixel Source

Une fois les coordonnées (u, v) calculées dans le référentiel du segment de destination PQ, le pixel source X' est reconstruit en appliquant ces mêmes coordonnées au segment source $P'Q'$:

$$X' = P' + u \cdot (Q' - P') + v \cdot \frac{\perp (Q' - P')}{\|Q' - P'\|} \tag{3.3}$$

Propriété 3.1 (Conservation des coordonnées locales)**.** *X et X' possèdent exactement les mêmes coordonnées (u, v) par rapport à leurs segments respectifs PQ et $P'Q'$. Le mapping $X \mapsto X'$ est un mapping affine local qui préserve les positions relatives.*

Cette propriété explique le comportement aux bornes : lorsque $t = 0$, les segments $M(t) = A$, donc $PQ = P'Q'$ et $X' = X$ (le warp est l'identité sur I_0). De même à $t = 1$.

3.3 Implémentation des Équations de Base

Listing 3.1 – BeierNeely.cpp — fonctions fondamentales

```
cv::Point2f perpendicular(const cv::Point2f& v) {
    return cv::Point2f(-v.y, v.x);
}

void uvFromLine(const cv::Point2f& X, const LineSeg& L,
                float& u, float& v) {
    cv::Point2f PQ  = L.Q - L.P;
    float len2 = PQ.dot(PQ);
    float len  = std::sqrt(std::max(len2, 1e-8f));
    cv::Point2f XP = X - L.P;
    u = (len2 < 1e-8f) ? 0.0f : (XP.dot(PQ) / len2);
    v = XP.dot(perpendicular(PQ)) / len;
}

cv::Point2f pointFromUV(const LineSeg& src, float u, float v) {
    cv::Point2f PQ = src.Q - src.P;
    float len = std::sqrt(std::max(PQ.dot(PQ), 1e-8f));
    return src.P + u * PQ + v * (perpendicular(PQ) / len);
}
```

4 Transformation Multi-Segments et Fonction de Poids

4.1 Généralisation au Cas Multi-Segments

En présence de N paires de segments, chaque segment i induit un déplacement candidat $D_i = X_i' - X$ pour le pixel X. Ces déplacements sont combinés par une moyenne pondérée :

$$X' = X + \frac{\sum_{i=1}^{N} D_i \cdot w_i}{\sum_{i=1}^{N} w_i} \tag{4.1}$$

où le poids w_i quantifie l'influence du segment i sur le pixel X en fonction de leur distance relative et de la longueur du segment.

4.2 La Fonction de Poids Originale

4.2.1 Définition

Beier et Neely définissent la fonction de poids suivante :

$$w_i = \left(\frac{\text{length}_i^p}{a + \text{dist}_i} \right)^b \tag{4.2}$$

où dist_i est la distance du pixel X au segment i, calculée selon la règle :

$$\text{dist}_i = \begin{cases} |v_i| & \text{si } 0 \leq u_i \leq 1 \\ \|X - P_i\| & \text{si } u_i < 0 \\ \|X - Q_i\| & \text{si } u_i > 1 \end{cases} \tag{4.3}$$

4.2.2 Rôle des Paramètres

Paramètre	Rôle	Effet sur le warp
dist_i	Distance de X au segment i	Perp. si $0 \leq u \leq 1$, distance à l'extrémité sinon
length_i	Longueur du segment i	Segments longs plus influents si $p > 0$
$a > 0$	Régularisation	Petit a : forte localité ; grand a : warp global doux
$b > 0$	Exposant de décroissance	Grand b : influence quasi-nulle à distance ; seul le segment le plus proche compte
$p \geq 0$	Exposant de longueur	$p = 0$: tous les segments égaux ; $p > 0$: segments longs dominants

TABLE 4.1 – Rôle des paramètres de la fonction de poids de Beier-Neely.

4.2.3 Décroissance Hyperbolique

La fonction de poids (4.2) présente une décroissance polynomiale en $1/d^b$ (pour $p = 0$ et a petit). Cette décroissance est relativement lente : les segments éloignés conservent une influence résiduelle non nulle sur l'ensemble de l'image.

4.3 Notre Variante : Fonction de Poids Exponentielle

4.3.1 Motivation et Apport

La décroissance polynomiale de la formule originale engendre une influence diffuse sur toute l'image, même pour des segments éloignés du pixel traité. En accord avec notre encadrant, nous proposons une variante remplaçant le terme $1/(a + d)$ par une décroissance exponentielle $e^{-k \cdot d}$:

$$w_i^{\text{exp}} = \text{length}_i^p \cdot e^{-k \cdot \text{dist}_i} \tag{4.4}$$

Cette formulation apporte deux avantages par rapport à l'existant : une coupure plus nette de l'influence à grande distance, et une réduction du nombre de paramètres à deux (k et p) au lieu de trois (a, b, p).

4.3.2 Comparaison Analytique

Propriété	**Originale** $\left(\frac{\text{len}^p}{a+d}\right)^b$	**Exponentielle** $\text{len}^p \cdot e^{-kd}$
Type de décroissance	Polynomiale en $1/d^b$	Exponentielle
Valeur en $d = 0$	$(1/a)^b$	1
Nombre de paramètres	3 (a, b, p)	2 (k, p)
Influence à l'infini	$\to 0$ lentement	$\to 0$ rapidement
Coupure nette	Non	Oui (grand k)

TABLE 4.2 – Comparaison analytique de la formule originale et de notre variante exponentielle.

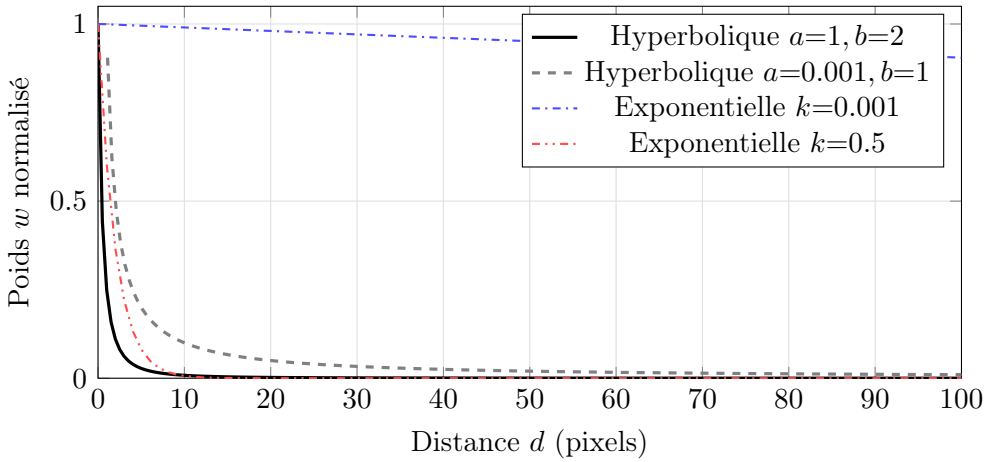


FIGURE 4.1 – Comparaison des fonctions de poids hyperbolique et exponentielle en fonction de la distance. k=0.001 produit une décroissance très lente (influence quasi-globale) ; k=0.5 impose une coupure très abrupte (influence strictement locale).

4.3.3 Implémentation de la Variante

La modification porte uniquement sur la fonction `weightForLine` dans `Warp.cpp`, ce qui illustre la modularité de l'architecture :

Listing 4.1 – Variante exponentielle dans Warp.cpp

```
// Formule originale Beier-Neely
static inline float weightForLine_original(float length, float dist,
                                           const BNParams& prm) {
    float lp = std::pow(std::max(length, 1e-8f), prm.p);
    return std::pow(lp / (prm.a + dist), prm.b);
}

// Notre variante exponentielle
static inline float weightForLine_exp(float length, float dist,
```

```
                                        const BNParams& prm) {
    float lp = std::pow(std::max(length, 1e-8f), prm.p);
    return lp * std::exp(-prm.k * dist);
}
```

La structure `BNParams` est augmentée du paramètre k propre à notre variante :

Listing 4.2 – Structure BNParams étendue

```
struct BNParams {
    float a = 1.0f;   // regularisation (formule originale)
    float b = 2.0f;   // exposant de decroissance (formule originale)
    float p = 0.0f;   // exposant de longueur (commun aux deux
        formules)
    float k = 0.05f;  // taux de decroissance exponentielle (notre
        variante)
};
```

5 Pseudo-codes Formels

5.1 Calcul des Coordonnées Locales

Algorithm 1 UvFromLine — Coordonnées locales (u, v)

Require: Pixel $X \in \mathbb{R}^2$, segment $L = (P, Q)$, $\varepsilon = 10^{-8}$
Ensure: Coordonnées réelles (u, v)

$\vec{d} \leftarrow Q - P$
$\ell^2 \leftarrow \vec{d} \cdot \vec{d}$
$\ell \leftarrow \sqrt{\max(\ell^2, \varepsilon)}$
$\vec{r} \leftarrow X - P$
if $\ell^2 < \varepsilon$ **then**
    $u \leftarrow 0$
else
    $u \leftarrow (\vec{r} \cdot \vec{d}) \,/\, \ell^2$
end if
$v \leftarrow (\vec{r} \cdot \perp (\vec{d})) \,/\, \ell$
return (u, v)

5.2 Reconstruction du Pixel Source

Algorithm 2 PointFromUV — Reconstruction de X'

Require: Coordonnées (u, v), segment source $L' = (P', Q')$
Ensure: Pixel source $X' \in \mathbb{R}^2$

$\vec{d'} \leftarrow Q' - P'$
$\ell' \leftarrow \sqrt{\max(\vec{d'} \cdot \vec{d'}, \varepsilon)}$
$\vec{n} \leftarrow \perp (\vec{d'}) \,/\, \ell'$
return $P' + u \cdot \vec{d'} + v \cdot \vec{n}$

5.3 Calcul de la Distance au Segment

Algorithm 3 DistToSeg — Distance de X au segment L

Require: Pixel X, segment $L = (P, Q)$, coordonnées (u, v)
Ensure: Distance $d \geq 0$

if $0 \leq u \leq 1$ **then**
    return $|v|$
else if $u < 0$ **then**
    return $\|X - P\|$
else
    return $\|X - Q\|$
end if

5.4 Warp d'un Pixel — Multi-Segments

Algorithm 4 WarpPixel — Calcul de X' par moyenne pondérée

Require: X, segments sources $\{L_i^{\text{src}}\}_{i=1}^{N}$, segments destination $\{L_i^{\text{dst}}\}_{i=1}^{N}$, paramètres (a, b, p)
Ensure: Coordonnées du pixel source $X' \in \mathbb{R}^2$

$\vec{S} \leftarrow \mathbf{0}$ ▷ somme pondérée des déplacements
$W \leftarrow 0$ ▷ somme des poids
for $i \leftarrow 1$ **à** N **do**
    $(u_i, v_i) \leftarrow \text{UvFromLine}(X, L_i^{\text{dst}})$
    $\hat{X}_i \leftarrow \text{PointFromUV}(L_i^{\text{src}}, u_i, v_i)$
    $D_i \leftarrow \hat{X}_i - X$
    $d_i \leftarrow \text{DistToSeg}(X, L_i^{\text{dst}}, u_i, v_i)$
    $\ell_i \leftarrow \|L_i^{\text{dst}}.Q - L_i^{\text{dst}}.P\|$
    $w_i \leftarrow \left(\ell_i^p / (a + d_i)\right)^b$
    $\vec{S} \leftarrow \vec{S} + w_i \cdot D_i$
    $W \leftarrow W + w_i$
end for
if $W > \varepsilon$ **then**
    return $X + \vec{S}/W$
else
    return X
end if

5.5 Warp Complet d'une Image

Algorithm 5 WarpImage — Déformation par reverse mapping

Require: Image source I, segments $\{L_i^{\text{src}}\}$, $\{L_i^{\text{dst}}\}$, paramètres
Ensure: Image déformée I'

$I' \leftarrow$ image nulle de même dimension que I
for $y \leftarrow 0$ **à** $H - 1$ **do**
    for $x \leftarrow 0$ **à** $W - 1$ **do**
        $X \leftarrow (x, y)$
        $X' \leftarrow \text{WarpPixel}(X, \{L^{\text{src}}\}, \{L^{\text{dst}}\}, \text{prm})$
        $I'[y][x] \leftarrow \text{BilinéaireSample}(I, X')$
    end for
end for
return I'

5.6 Morphing d'une Frame

Algorithm 6 MorphFrame — Calcul de $F(t)$

Require: Images I_A, I_B, paires $\{(A_i, B_i)\}_{i=1}^{N}$, $t \in [0,1]$, paramètres
Ensure: Frame interpolée $F(t)$

for $i \leftarrow 1$ **à** N **do** ▷ Interpolation des segments
    $M_i \leftarrow (1-t) \cdot A_i + t \cdot B_i$
end for
$W_A \leftarrow$ WarpImage(I_A, $\{A_i\}$, $\{M_i\}$, prm)
$W_B \leftarrow$ WarpImage(I_B, $\{B_i\}$, $\{M_i\}$, prm)
return $(1-t) \cdot W_A + t \cdot W_B$ ▷ Cross-dissolve

5.7 Détection Automatique des Landmarks Faciaux

Algorithm 7 DetectLandmarks — 68 points faciaux via dlib

Require: Image I, chemin du modèle `modelPath`
Ensure: Vecteur $\{p_i\}_{i=0}^{67}$ de points $\mathbb{R}^2$

Convertir I de BGR (OpenCV) vers RGB (dlib)
det $\leftarrow$ FrontalFaceDetector()
pred $\leftarrow$ LoadShapePredictor(`modelPath`)
faces $\leftarrow$ det(I)
if faces $= \emptyset$ **then**
    return $\emptyset$
end if
shape $\leftarrow$ pred(I, faces[0])
for $i \leftarrow 0$ **à** 67 **do**
    $p_i \leftarrow$ (shape.part(i).x, shape.part(i).y)
end for
return $\{p_i\}$

Algorithm 8 LandmarksToSegments — Conversion landmarks → paires

Require: Landmarks $\{p_i^A\}_{i=0}^{67}$ de I_A, $\{p_i^B\}_{i=0}^{67}$ de I_B
Ensure: 67 paires de segments $\{(A_i, B_i)\}_{i=0}^{66}$

for $i \leftarrow 0$ **à** 66 **do**
    $A_i \leftarrow (p_i^A,\ p_{i+1}^A)$
    $B_i \leftarrow (p_i^B,\ p_{i+1}^B)$
end for
return $\{(A_i, B_i)\}$

6 Implémentation C++/OpenCV

6.1 Architecture Modulaire

L'implémentation est organisée en cinq modules indépendants aux responsabilités bien délimitées.

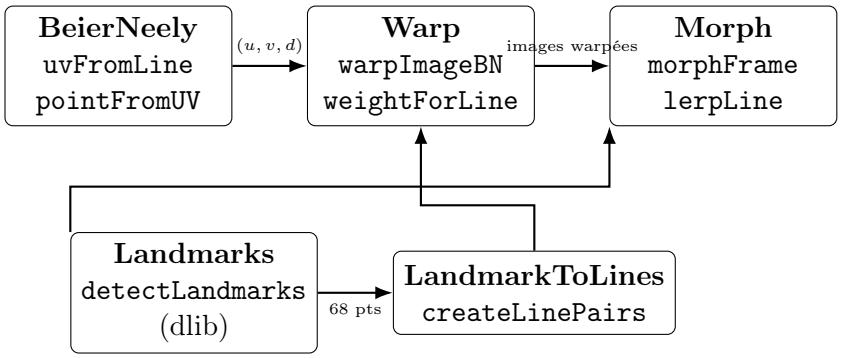


FIGURE 6.1 – Architecture des cinq modules C++ et flux de données.

Module	Rôle	Fonctions principales
`BeierNeely`	Cœur mathématique	`uvFromLine()`, `pointFromUV()`, `distPointToDirectedSeg()`
`Warp`	Déformation image	`warpImageBN()` — parcourt les pixels, calcule X', interpolation bi-linéaire
`Morph`	Orchestration	`morphFrame()` — interpole $M(t)$, appelle le warp, cross-dissolve
`Landmarks`	Détection faciale	dlib `frontal_face_detector` + `shape_predictor` (68 pts)
`LandmarkToLines`	Conversion	68 landmarks → 67 paires de segments consécutifs

TABLE 6.1 – Description des cinq modules de l'implémentation.

6.2 Module Morph.cpp

Listing 6.1 – Morph.cpp — orchestration du morphing

```
static inline LineSeg lerpLine(const LineSeg& L0,
                               const LineSeg& L1, float t) {
    LineSeg out;
    out.P = (1.0f - t) * L0.P + t * L1.P;
    out.Q = (1.0f - t) * L0.Q + t * L1.Q;
    return out;
```

```
}

cv::Mat morphFrame(const cv::Mat& imgA, const cv::Mat& imgB,
                   const std::vector<LinePair>& pairs,
                   float t, const BNParams& prm)
{
    std::vector<LineSeg> A, B, M;
    for (auto& lp : pairs) {
        A.push_back(lp.A);
        B.push_back(lp.B);
        M.push_back(lerpLine(lp.A, lp.B, t));
    }
    cv::Mat warpA = warpImageBN(imgA, A, M, prm);
    cv::Mat warpB = warpImageBN(imgB, B, M, prm);
    cv::Mat out;
    cv::addWeighted(warpA, 1.0 - t, warpB, t, 0.0, out);
    return out;
}
```

6.3 Interpolation Bilinéaire

Après le calcul de $X' = (x', y')$, la couleur est obtenue par interpolation bilinéaire sur les quatre pixels voisins entiers :

$$c(X') = (1 - f_x)(1 - f_y)\, c_{00} + f_x(1 - f_y)\, c_{10} + (1 - f_x) f_y\, c_{01} + f_x f_y\, c_{11} \tag{6.1}$$

où $f_x = x' - \lfloor x' \rfloor$, $f_y = y' - \lfloor y' \rfloor$, et c_{ij} désigne la couleur du pixel voisin $(x_0 + j, y_0 + i)$.

6.4 Détection Automatique des Landmarks Faciaux

La bibliothèque dlib [4] fournit un détecteur de visage frontal et un prédicteur de forme entraîné sur 68 points anatomiques standardisés [5]. Notre contribution consiste à intégrer cette détection pour remplacer intégralement le placement manuel des segments. Les points sont regroupés par régions :

Indices	Région	Nb de points
0 – 16	Contour du visage	17
17 – 26	Sourcils (gauche + droit)	10
27 – 35	Arête et pointe nasale	9
36 – 47	Yeux (gauche + droit)	12
48 – 67	Bouche et lèvres	20
Total		**68**

TABLE 6.2 – Groupes de landmarks faciaux du modèle dlib.

7 Analyse des Paramètres et Discussion

7.1 Complexité Algorithmique

La complexité du warp est en $\mathcal{O}(W \times H \times N)$, où $W \times H$ est la résolution de l'image et N le nombre de segments. Pour 67 segments dlib sur une image 512×512, cela représente environ $17{,}6 \times 10^6$ opérations élémentaires par frame.

7.2 Influence des Paramètres

7.2.1 Paramètre a — Portée de l'influence

Pour a petit, la fonction de poids approche ℓ^p/d^b : les segments très proches de X dominent massivement la somme. Pour a grand, tous les segments contribuent de façon plus uniforme, produisant un warp globalement doux mais moins précis géographiquement.

7.2.2 Paramètre b — Localité de l'influence

b contrôle la vitesse de décroissance du poids avec la distance. Pour b élevé (ex. $b = 4$), seul le segment immédiatement voisin d'un pixel l'influence significativement. Pour b faible ($b = 1$), plusieurs segments contribuent, produisant un effet de lissage global plus apparent.

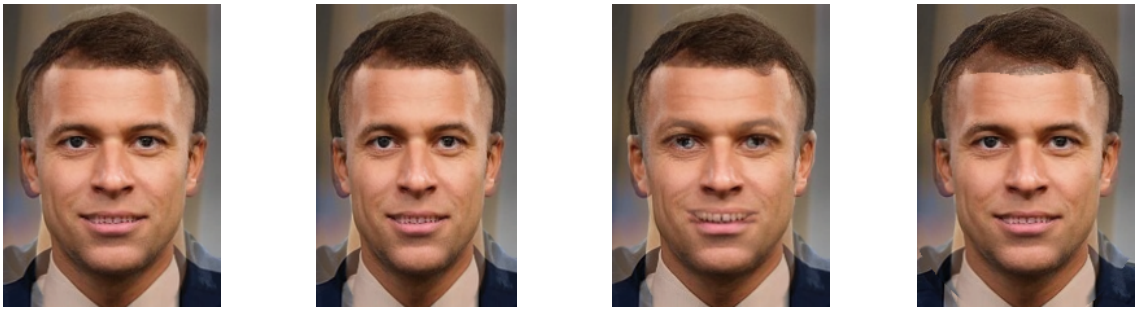
7.2.3 Paramètre k (notre variante exponentielle)

Le paramètre k joue un rôle analogue à b dans la formule originale :

- $k = 0.001$: décroissance très lente, influence quasi-globale.
- $k = 0.05$: décroissance modérée, comportement comparable à $a = 1, b = 2$.
- $k = 0.5$: coupure très abrupte, influence strictement locale.

7.3 Résultats Expérimentaux

Les figures 7.1a–7.1d présentent des frames intermédiaires ($t = 0.5$) obtenues avec différents paramètres sur une paire d'images faciales.



(a) Formule originale $a = 0.001,\ b = 1$

(b) Formule originale $a = 1,\ b = 2$

(c) Notre variante $k = 0.001$

(d) Notre variante $k = 0.5$

FIGURE 7.1 – Frames intermédiaires ($t = 0.5$) pour différents paramètres.

7.4 Propriété de Normalisation

Remarque 7.1. *La division par $\sum_i w_i$ dans l'équation (4.1) normalise automatiquement les déplacements, quelle que soit l'amplitude des poids.*

7.5 Comparaison avec d'Autres Techniques de Morphing

Technique	Contrôle	Qualité	Complexité	Automatisation
Cross-dissolve	Aucun	Faible	$\mathcal{O}(WH)$	Oui
Mesh warping (Wolberg)	Grille	Moyen	$\mathcal{O}(WHN)$	Partielle
Beier-Neely	Segments	Élevé	$\mathcal{O}(WHN)$	Semi (dlib)
Triangulation Delaunay	Points	Très élevé	$\mathcal{O}(WH + N \log N)$	Oui

TABLE 7.1 – Comparaison des principales techniques de morphing.

8 Triangulation de Delaunay et Algorithme de Bowyer-Watson

La triangulation de Delaunay constitue une amélioration structurelle de la méthode de Beier et Neely : au lieu de segments directeurs à influence globale pondérée, on partitionne l'image en triangles couvrant tout le domaine, chacun traité comme une transformation affine locale indépendante.

Définition 8.1 (Triangulation de Delaunay)**.** *La* triangulation de Delaunay *d'un ensemble de points P est l'unique triangulation telle que, pour tout triangle T, aucun point de P ne se trouve à l'intérieur du cercle circonscrit à T. Elle maximise le plus petit angle de tous les triangles, produisant des mailles régulières et évitant les triangles aplatis néfastes aux transformations affines.*

8.1 Algorithme de Bowyer-Watson

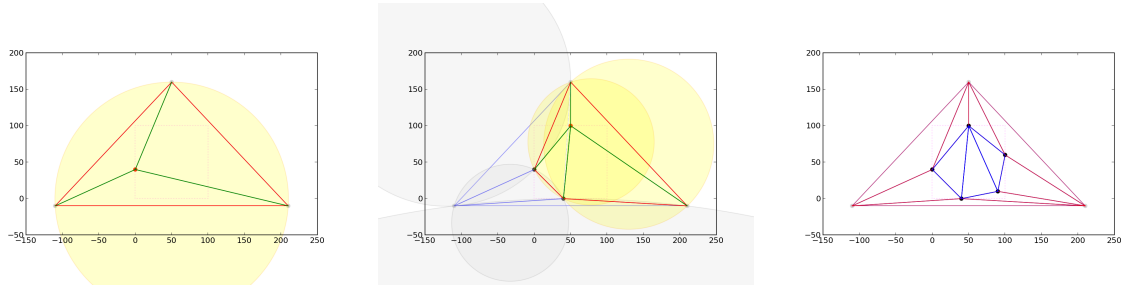
L'algorithme insère les points de P un à un et répare localement la triangulation à chaque étape :

1. **Super-triangle** : initialiser $\mathcal{T}$ avec un triangle englobant tous les points.
2. **Pour chaque point** p : identifier $\mathcal{B}(p)$, l'ensemble des triangles dont le cercle circonscrit contient p (triangles invalides).
3. **Reconstruction** : supprimer $\mathcal{B}(p)$, relier p aux arêtes frontières du trou formé.
4. **Nettoyage** : supprimer les triangles partageant un sommet avec le super-triangle.

Le test d'appartenance au cercle circonscrit du triangle ABC est donné par le signe du déterminant :

$$\Delta = \begin{vmatrix} A_x - p_x & A_y - p_y & (A_x^2 + A_y^2) - (p_x^2 + p_y^2) \\ B_x - p_x & B_y - p_y & (B_x^2 + B_y^2) - (p_x^2 + p_y^2) \\ C_x - p_x & C_y - p_y & (C_x^2 + C_y^2) - (p_x^2 + p_y^2) \end{vmatrix} > 0 \tag{8.1}$$

Les figures 8.1 illustrent la progression de l'algorithme sur cinq points, de l'insertion initiale jusqu'à la triangulation finale.



(a) Insertion du 1er point. Triangle rouge avant, vert après.

(b) Insertion du 3e point. Plusieurs triangles invalides détectés.

(c) Résultat final après suppression du super-triangle.

FIGURE 8.1 – Progression de l'algorithme de Bowyer-Watson. *Source* : https://fr.wikipedia.org/wiki/Algorithme_de_Bowyer-Watson

8.2 Application au Morphing par Coordonnées Barycentriques

On construit deux triangulations $\mathcal{T}_A$ et $\mathcal{T}_B$ de même topologie sur les images source et cible à partir des mêmes landmarks dlib. Pour chaque pixel X de l'image morphée, le *backward mapping* s'effectue en trois étapes :

1. Localiser X dans le triangle $T_i^M(t) = (1-t)\,T_i^A + t\,T_i^B$.
2. Calculer ses coordonnées barycentriques (α, β, γ) : $X = \alpha A_M + \beta B_M + \gamma C_M$, avec $\alpha + \beta + \gamma = 1$.
3. Reconstruire le pixel source : $X' = \alpha A_A + \beta B_A + \gamma C_A$.

La couleur finale est : $F(t)[X] = (1-t) \cdot c(I_A, X'_A) + t \cdot c(I_B, X'_B)$.

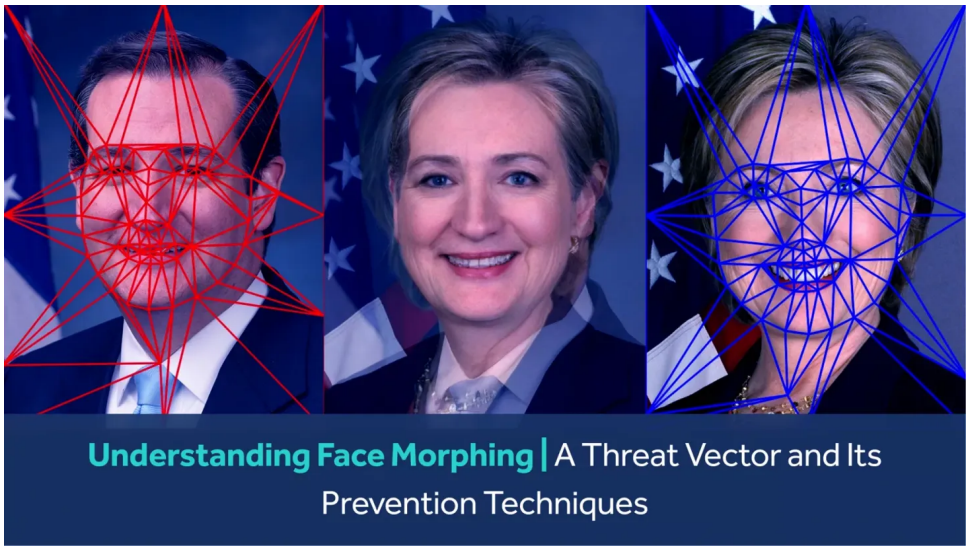


FIGURE 8.2 – Triangulations de Delaunay source (rouge) et destination (bleu) sur deux visages. Même topologie, correspondance triangle à triangle. *Source* : https://facia.ai/blog/understanding-face-morphing-a-threat-vector-and-its-prevention-techniques/

Cette approche garantit une correspondance *exacte et locale* (contrairement à la moyenne pondérée globale de Beier-Neely), supprime les artefacts aux jonctions d'influence, et réduit la complexité à $\mathcal{O}(WH + N \log N)$.

9 Conclusion

Nous avons présenté une étude approfondie de l'algorithme de field morphing de Beier et Neely, depuis ses fondements mathématiques jusqu'à une implémentation modulaire en C++/OpenCV, en y apportant plusieurs contributions originales.

Sur le plan théorique, nous avons formalisé les trois équations centrales — coordonnées (u, v), reconstruction de X', moyenne pondérée multi-segments — et justifié le recours au reverse mapping par ses propriétés de surjectivité.

Sur le plan contributif, nous avons proposé une variante à décroissance exponentielle réduisant le nombre de paramètres de trois à deux, et intégré dlib pour automatiser entièrement le placement des segments. Enfin, le chapitre 8 ouvre la perspective d'une amélioration par triangulation de Delaunay : en remplaçant la moyenne pondérée globale par un backward mapping barycentrique triangle par triangle, cette approche supprimerait les artefacts aux jonctions d'influence et réduirait la complexité de $\mathcal{O}(WHN)$ à $\mathcal{O}(WH + N \log N)$.

Équations fondamentales de l'algorithme :

$$u_i = \frac{(X - P_i) \cdot (Q_i - P_i)}{\|Q_i - P_i\|^2}, \qquad v_i = \frac{(X - P_i) \cdot \perp (Q_i - P_i)}{\|Q_i - P_i\|}$$

$$X'_i = P'_i + u_i(Q'_i - P'_i) + v_i \frac{\perp (Q'_i - P'_i)}{\|Q'_i - P'_i\|}$$

$$X' = X + \frac{\sum_i (X'_i - X) \cdot w_i}{\sum_i w_i}$$

$$w_i = \left(\frac{\text{length}_i^p}{a + \text{dist}_i} \right)^b \quad \text{(original)} \qquad w_i^{\exp} = \text{length}_i^p \cdot e^{-k\,\text{dist}_i} \quad \text{(notre variante)}$$

Bibliographie

[1] T. Beier and S. Neely, *Feature-Based Image Metamorphosis*, in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*, ACM, New York, vol. 26, no. 2, pp. 35–42, July 1992.

[2] G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1990, 318 p.

[3] G. Bradski, *The OpenCV Library*, *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 122–125, November 2000.

[4] D. E. King, *Dlib-ml : A Machine Learning Toolkit*, *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1755–1758, 2009.

[5] V. Kazemi and J. Sullivan, *One Millisecond Face Alignment with an Ensemble of Regression Trees*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, IEEE, Columbus, OH, USA, pp. 1867–1874, 2014.

[6] A. Bowyer, *Computing Dirichlet Tessellations*, *The Computer Journal*, vol. 24, no. 2, pp. 162–166, 1981.

[7] D. F. Watson, *Computing the n-dimensional Delaunay Tessellation with Application to Voronoi Polytopes*, *The Computer Journal*, vol. 24, no. 2, pp. 167–172, 1981.

A Historique des Réunions avec l'Encadrant

Date	**Ordre du jour / Points abordés**
5 février 2026 10h00	Introduction au morphing d'images : définition générale, étymologie et domaines d'application. Présentation des objectifs du projet : réalisation d'une implémentation en C++, exploration de méthodes d'amélioration, avec pour référence visuelle la séquence de morphing facial du clip *Black or White* de Michael Jackson (1991). Choix de l'article de référence : T. Beier and S. Neely, *Feature-Based Image Metamorphosis*, SIGGRAPH '92.
16 mars 2026 9h45	Présentation d'une première réalisation du projet. Discussion des étapes nécessaires à la transition de morphing : warp avec backward mapping (reverse mapping), détection des landmarks faciaux, calcul de la correspondance par une paire de segments (coordonnées u, v), et généralisation à plusieurs paires de segments par moyenne pondérée.
8 avril 2026	Revue de l'exécution du projet et de son architecture logicielle (5 modules C++). Discussion sur l'amélioration de la présentation pour couvrir l'ensemble des points importants du projet. Proposition d'une amélioration du morphing par intégration de la triangulation de Delaunay comme alternative aux segments directeurs.
4 mai 2026 10h00	Discussion approfondie sur l'importance de la formule mathématique de Beier & Neely et analyse des conséquences d'une modification de la fonction de poids (proposition de la variante exponentielle). Discussion sur une meilleure représentation schématique de la transformation par une paire de segments. Comparaison de la méthode de Beier-Neely avec d'autres techniques de morphing (cross-dissolve, mesh warping, thin-plate spline, optical flow).
12 mai 2026 10h00	Présentation détaillée de l'intégration potentielle de la triangulation de Delaunay dans le projet de morphing. Discussion sur l'algorithme de Bowyer–Watson pour la construction incrémentale de la triangulation et sur son rôle dans l'amélioration des déformations locales. Vérification et amélioration des supports de présentation.(réunion prévue)

TABLE A.1 – Historique des réunions effectuées et prévues avec l'encadrant du sujet.

B Sujet Original

Master 1ère année. Année 2025-2026 Novembre 2025
Sujet de I.R. de D. Michel
domic62@hotmail.com
LGIPM - UFR MIM

Les techniques de morphing

Sujet :

Le morphing est une technique de déformation d'image utilisée en synthèse d'images. Tout le monde a déjà vu des applications de ces techniques dans des animations, des films, des pubs, des clips (dont le célèbre *Black Or White* de Michael Jackson en 1991), etc. Le morphing peut être réalisé en 3D ou en 2D, sur des images de type bitmap ou vectoriel. Le but de ce sujet est double. Il s'agit d'abord d'établir, du point de vue algorithmique, un état de l'art de toutes les techniques de morphing. Puis, cet état de l'art effectué, un des algorithmes étudié sera retenu pour être mis en oeuvre. Pour simplifier, on pourra se limiter à la réalisation d'une maquette 2D. Les aspects géométriques et algorithmiques seront largement discutés avec l'encadrant.

Nombre d'étudiants : 2

Implémentation

L'implémentation sera réalisée en JAVA ou en C++

Mots clés

morphing, interpolation linéaire, combinaison linéaire convexe, warping, cinématique, Beier-Neely algorithm, traitement d'image, blending, fondu