travail sur la chromakey

Les méthodes exposées ci-après correspondent à une phase de développement - recherche qui s'inscrit dans le cadre du projet Rotoscopie. Nous renvoyons le lecteur aux rapports correspondants.

Le prototype s'appuie sur un logiciel créé précédemment à REALViZ SA : Image Viewer.

1) Introduction:

La plupart des approches de « segmentation-tracking » n'utilise pas d'information sur la couleur, et s'appuie essentiellement sur la luminance, c'est à dire l'intensité de l'image en niveau de gris. La raison majeure de ce choix est la très forte sensibilité de la couleur à l'illumination : développer une méthode robuste qui tienne compte de cette information n'est pas chose aisée. Cependant, dans le cadre de taches telles que la rotoscopie, une prise en compte de toute l'information disponible nous paraît essentielle.

Dans cette première phase de recherche, l'étendue des connaissances sur la chromakey, applicables a la rotoscopie, a été parcourue, et plusieurs méthodes ont été développées afin d'améliorer les résultats de « segmentation-tracking ». Notamment, des méthodes d'estimation de l'alpha-channel et de compositing, ainsi que des outils de post-rotoscopie ont été implémentés, des outils d'aide à la segmentation sont en phase de développement, et plusieurs pistes de « développement-recherche » sont toujours en phase d'étude.

Dans cette note, nous allons tout d'abord exposer les différentes techniques proposées et utilisées, avant de montrer les résultats auxquels nous sommes actuellement déjà parvenus.

2) Explications techniques :

a) Alpha-Channel estimation et Compositing :

ALPHA CHANNEL ESTIMATION

Il convient tout d'abord de faire remarquer que l'estimation du coefficient de compositing entre deux images, autrement appelé alpha-channel, est un *problème mal posé qui ne possède pas, en général, une solution unique*. Toute la difficulté est donc l'élaboration d'une méthode proposant une solution vraisemblable, dont les résultats perceptibles sont acceptables.

Cette méthode s'appuie sur celle exposée par Mark A.Ruzon et Carlo Tomasi [1]. Nous cherchons à exprimer alpha, la proportion avec laquelle deux couleurs se trouvent 'mixées' à la frontière d'un objet, comme un coefficient maximisant une probabilité sur un espace de couleur donné (ou plutôt devrait t-on dire une distribution, puisque nous travaillons avec la fonction de répartition ; cependant on ne s'offusquera pas de l'utilisation de ce terme).

Pour chaque pixel situé à la frontière, nous supposons que la couleur apparente est le résultat d'un morphing entre plusieurs couleurs provenant d'une zone extérieure et d'une zone intérieure. Chaque couleur est modélisée par une distribution gaussienne tridimensionnelle (dans un soucis d'optimisation, notamment pour l'intégration de cette approche dans la rotoscopie proprement-

dite, il est possible, sans que les résultats ne soient trop altérés, de travailler avec des distributions gaussiennes unidimensionnelles).

Trouver la proportion alpha d'un pixel de couleur $\vec{c}$, qui est supposée mixer correctement les distributions gaussiennes, revient à trouver le coefficient de morphing qui maximise en $\vec{c}$ une probabilité (distribution) définie sur un graphe d'affinité précis.

i) Elaboration d'une signature :

Etant donné un ensemble de pixels appartenant au background (définissant la zone extérieure que l'on souhaite « supprimer ») et au foreground (définissant la zone intérieure que l'on souhaite « conserver »), il est possible de construire deux signatures, une extérieure et une intérieure, représentant les différentes composantes gaussiennes présentes dans chaque sélection (voir l'annexe I pour la définition d'une composante gaussienne). Notons que l'espace dans lequel les composantes sont calculées ne correspond pas a l'espace RGB classique, mais a l'espace CIE Lab, plus approprié pour les calculs mathématiques (voir l'annexe II).

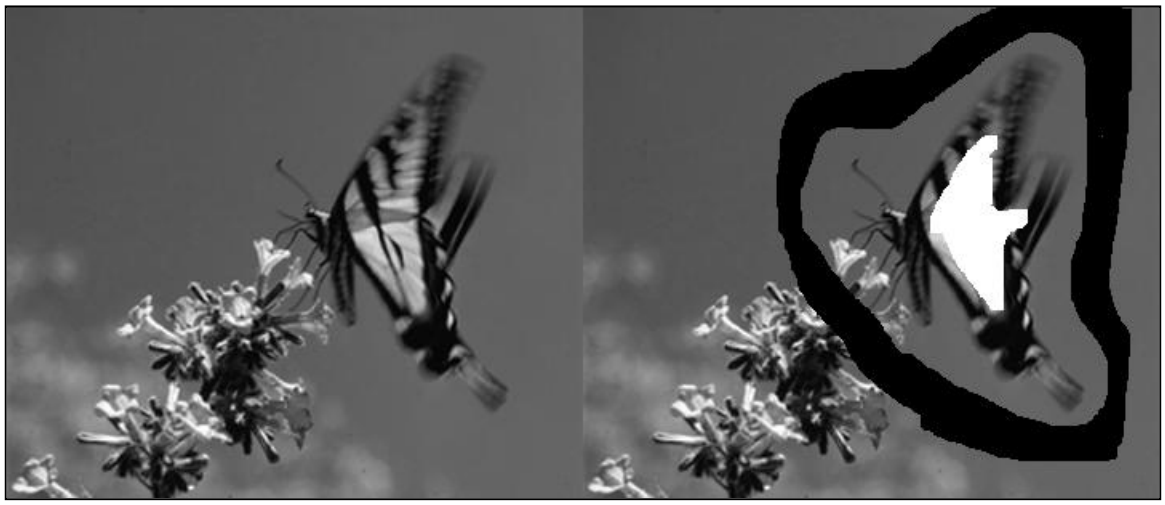


Figure 1: image originale, sélection de pixels intérieurs et extérieurs

Nous suivons la méthode de Michael Orchard & Charles Bouman [2] pour extraire d'un ensemble de pixels les composantes principales gaussiennes, en modifiant très légèrement les conditions de « splitting » d'un nœud.. Cette méthode est relativement simple et rapide. La partition d'un ensemble de pixels possède la structure d'un arbre binaire. Chaque feuille de l'arbre regroupe un sous-ensemble de pixels définissant une composante gaussienne. Ainsi, pour chaque feuille, la moyenne et la matrice de covariance sont calculées.

Pour construire l'arbre binaire, il s'agit de comprendre comment une feuille, c'est à dire un ensemble de pixels, est « séparée » en deux sous-partitions. On calcule le vecteur propre (de la matrice de covariance) associé à la valeur propre principale : cela donne la direction de l'ensemble du cluster qui maximise la dispersion. Perpendiculairement à cette direction, on « sépare » alors l'ensemble de pixels en deux sous-ensembles (tout simplement avec une distance signée), qui vont alors définir deux nouvelles composantes gaussiennes.
Notons que si la valeur propre principale est inférieure à 10 unités CIE Lab, nous gardons tel quel l'ensemble de pixels. Cela permet d'éviter de construire une signature avec un nombre trop important de composantes voisines. Enfin, la densité locale, correspondante à la densité au centre

du cluster (probabilité *38 %*, soit *demi-variance*) est également évaluée : si cette dernière est inférieure a *20%*, la feuille est automatiquement « séparée » en deux sous-feuilles.

On peut alors définir l'algorithme de séparation en composantes principales gaussiennes. A la première étape, l'ensemble des pixels permet l'évaluation d'une première moyenne et matrice de covariance. Suivant les conditions exposées précédemment, deux nouveaux nœuds sont construits.

Ensuite il s'agit de trouver la feuille possédant la valeur propre maximale (ce qui doit logiquement représenter l'ensemble avec la plus grande dispersion), et de « splitter » ce nœud. On recommence l'opération jusqu' à ce que

i) toutes les feuilles remplissent les conditions précédentes pour définir une composante gaussienne, ou bien
ii) le nombre de composantes est supérieure a *M* (On se limite à *M=10* composantes gaussiennes car cette approche est une approche locale, et un nombre important de composantes est tout simplement aberrant ; logiquement, *2-3* composantes devraient être un nombre suffisant, mais dans le but de pouvoir appliquer cette approche globalement (pour justement montrer ses limites), nous avons choisi un nombre plus élevé.

Complexité : A chaque étape, la construction d'un nœud est proportionnelle au nombre de pixels composant ce nœud. Sachant que le nombre maximal de composantes est *10*, nous obtenons une complexité linéaire en le nombre de pixels de départ. Une évaluation un peu plus précise donne une complexité de ~50 N.

Résultats :

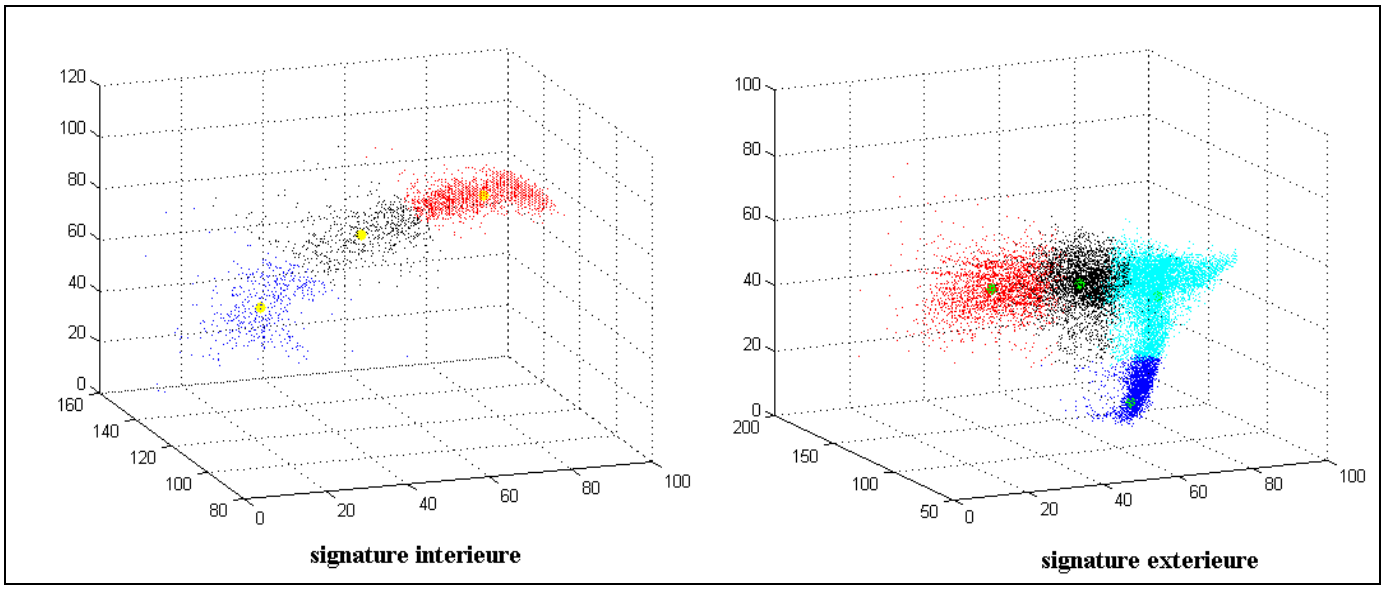


Figure 2: la représentation des différents sous-ensembles de pixels

Figure 3: image originale, vectorisation en 9 composantes, vectorisation en 18 composantes

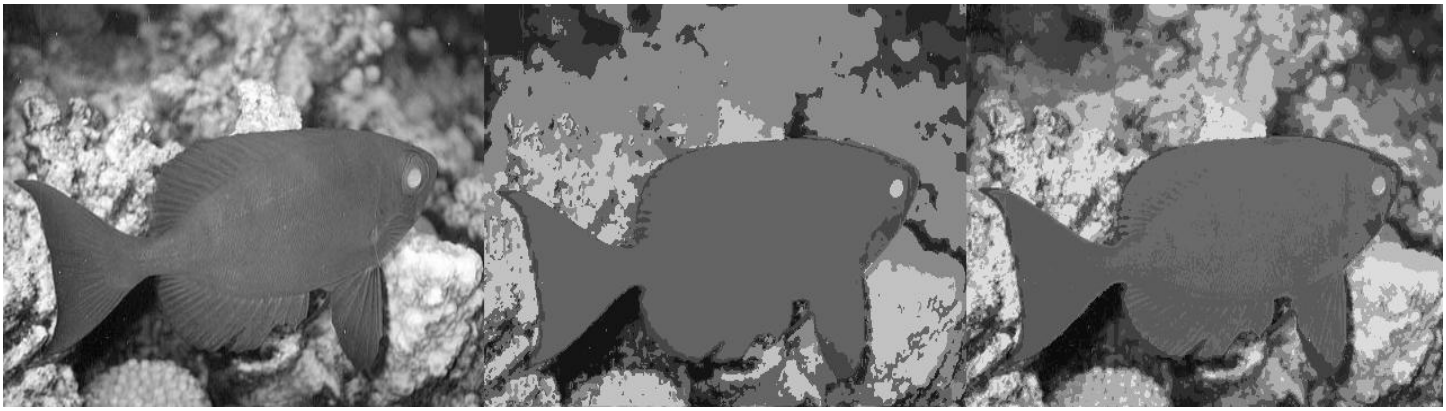


Figure 4: image originale, vectorisation en 9 composantes, vectorisation en 25 composantes

Cette première passe permet d'obtenir deux représentations simplifiées de l'extérieur et de l'intérieur de l'objet.

ii) Construction d'un graphe :

Un graphe est construit dans l'espace de couleur CIE Lab. Il représente les affinités entre couleurs dans les espaces de signatures opposées. Ce graphe est l'approche qui permet de « redéfinir » le précédent problème mal posé, et de lui trouver une solution unique. Dans cette approche, une affinité (entre deux composantes gaussiennes) est dite ambiguë si un point de l'espace de couleur peut être le résultat d'une ou plusieurs autres affinités.

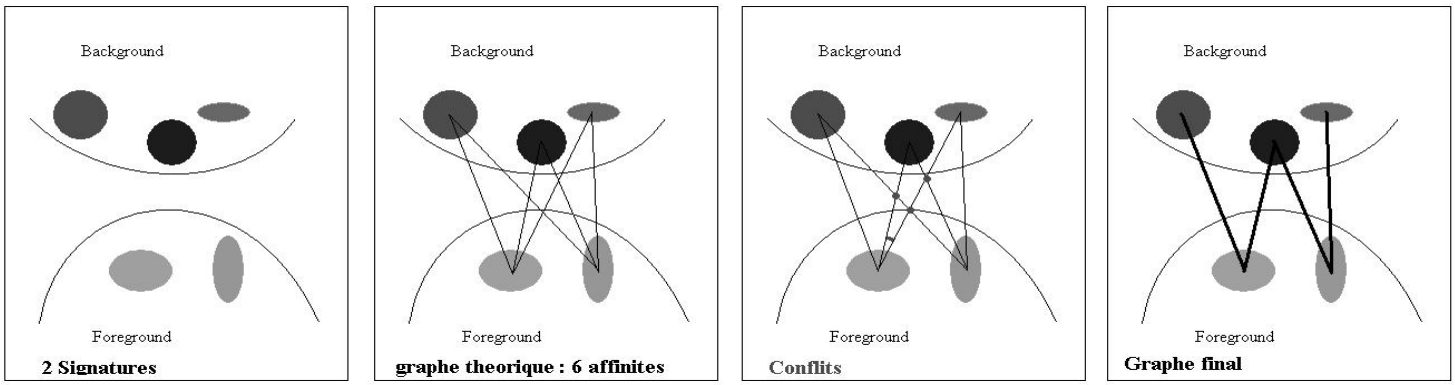
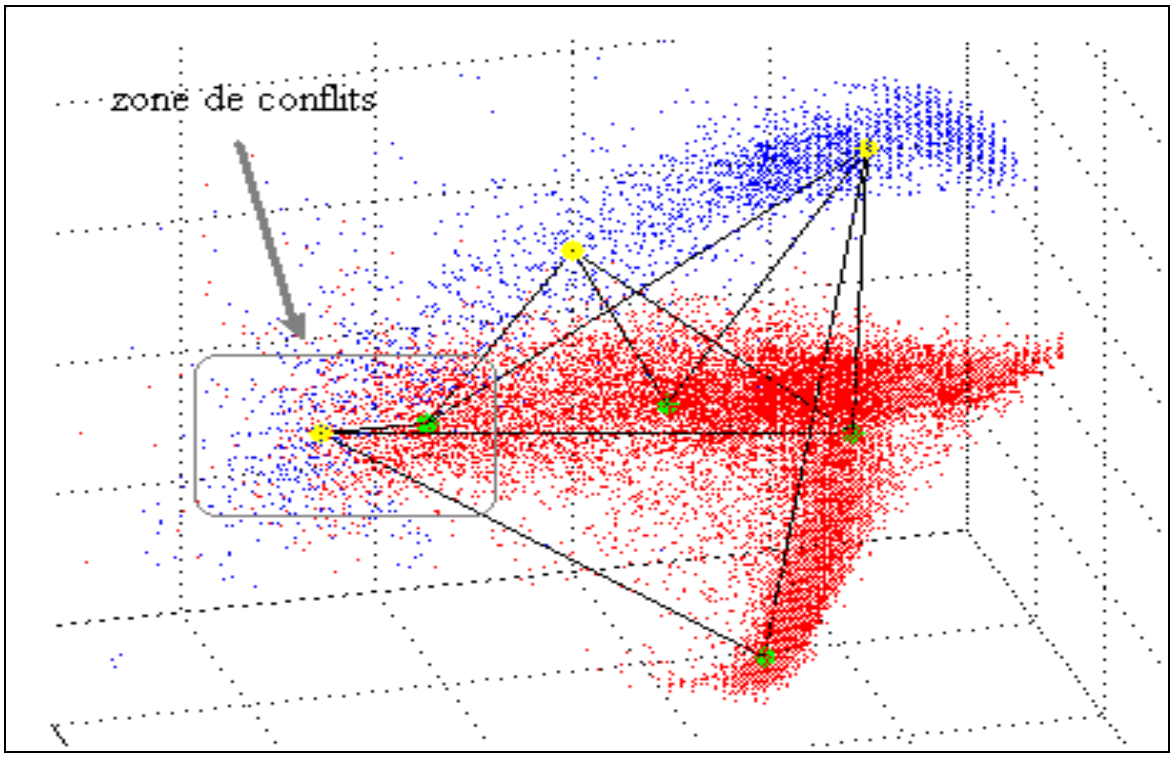


Figure 5: exemple de graphe en 2D (pour simplification)

L'ambiguïté peut résulter de deux types de conflits : conflit d'intersection ou conflit d'angle. On dira qu'il y a ambiguïté lorsque la distance entre deux segments est inférieure a 5 unités (CIE Lab), ou lorsque l'angle entre deux segments est inférieur a 1 degré (toujours dans l'espace CIE Lab). Dans les cas de conflits le segment le plus long est toujours le segment supprimé du graphe.

Dans le cas de notre exemple précédent, nous obtenons un graphe avec dix branches, au lieu d'un graphe théorique a *3*4 = 12* branches.



Dans l'exemple choisi, la zone de conflits résulte de la sélection de pixels verts et noirs appartenant aux fleurs qui se mixent avec les pixels noirs du papillon. Nous constatons

alors les limites de l'approche globale, limites qui seront évitées dans une approche plus locale comme nous le verrons un peu plus loin.

iii) Définition d'une probabilité :

A partir de ce graphe, il est facile de définir une probabilité sur l'ensemble de l'espace couleur pour une mixture alpha donnée. Définissons tout d'abord la probabilité pour une mixture *t* donnée:

$p_t(\vec{c}) = \sum_{k=1}^{n} a_k \quad G_k(\vec{c}, \mu_k(t), \mathrm{cov}_k(t))$, ou *n* est le nombre d'arrêtes dans le graphe, avec

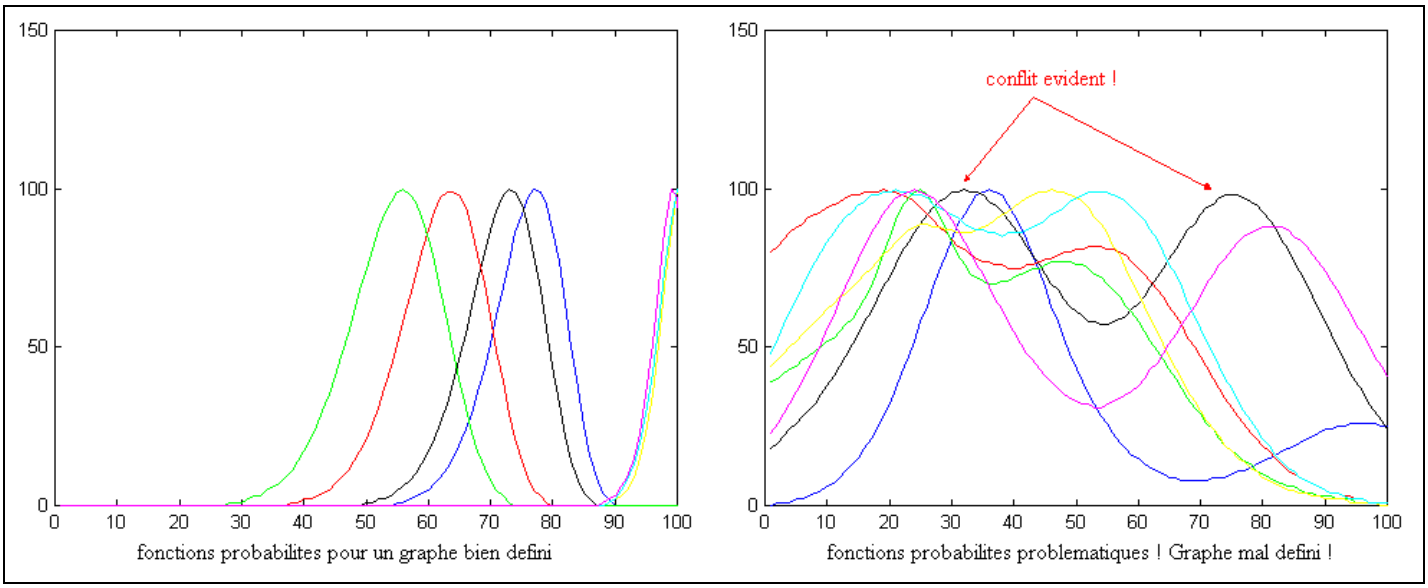
$\begin{cases} \vec{\mu}_k(t) = (1-t)\overrightarrow{col}_{ext} + t\overrightarrow{col}_{\mathrm{int}} \\ \underline{\underline{\mathrm{cov}}}_k(t) = (1-t)\underline{\underline{\mathrm{cov}}}_{ext} + t\underline{\underline{\mathrm{cov}}}_{\mathrm{int}} \end{cases}$, et a_k est théoriquement proportionnel aux pourcentages respectifs des deux couleurs étant les extrémités du segment *k*. G_k est bien entendu une fonction gaussienne. En réalité, nous avons choisi de prendre a_k constant, puisque la proportionnalité aux pourcentages respectifs de couleurs nous paraît peu pertinente.

Cette définition est relativement obscure, et il convient de l'expliciter un peu. Pour *t=0,* la probabilité correspond tout simplement à une somme des composantes gaussiennes extérieures. Pour *t=1,* c'est une somme des composantes intérieures uniquement. Pour une valeur de *t* quelconque, comprise entre *0* et *1,* cela correspond à une mixture des deux signatures. Ainsi lorsque *t* varie de *0* a *1,* nous obtenons un morphing des composantes extérieures dans les composantes intérieures.

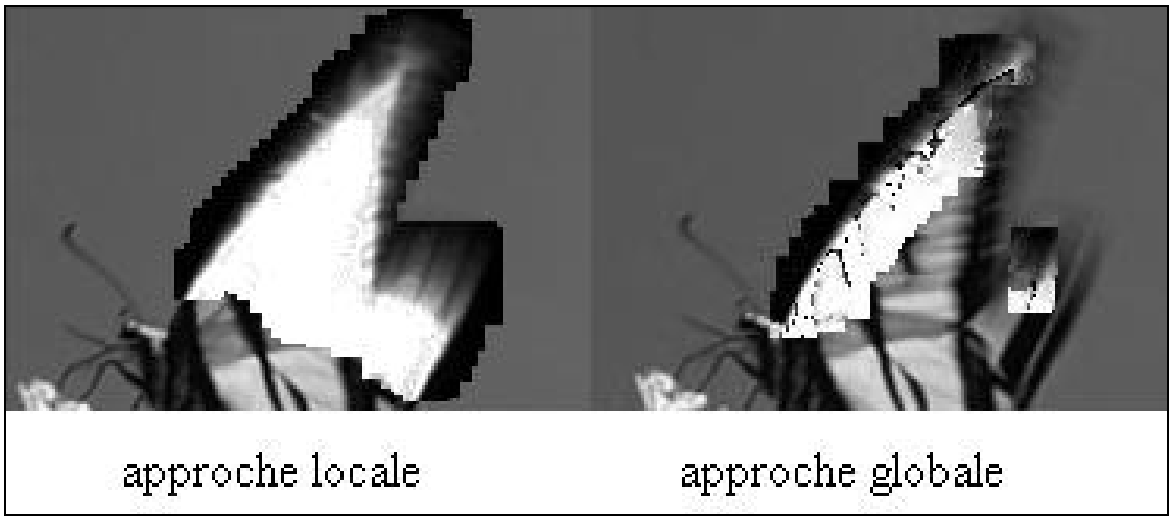
iv) Obtention du coefficient alpha :

Pour chaque pixel de couleur $\vec{c}$, il suffit d'évaluer $p_t(\vec{c})$ pour *t* variant de 0 a 1. Le coefficient alpha est la valeur de *t* maximisant cette fonction p_t. Notons que cette fonction n'est malheureusement pas toujours convexe, ce qui ne permet pas l'utilisation d'algorithmes efficaces.

L'allure générale de cette fonction est généralement « agréable », possédant un unique maximum bien que non convexe, notamment lorsque le graphe est défini sans ambiguïtés. Cependant, un graphe conflictuel, ainsi que certains pixels ambigus peuvent donner des fonctions nettement moins « agréables », pour lesquelles le maximum n'indique pas forcement le bon coefficient alpha !



Nous obtenons alors des résultats qui peuvent être totalement différents suivants que l'on se place dans une approche locale ou bien dans une approche globale.



COMPOSITING: METHODE BAYESIENNE

L'estimation du coefficient alpha à partir de la construction des signatures extérieure et intérieure permet d'estimer de manière très précise l'image « foreground » ainsi que de la recomposer directement avec une autre image.

Pour chaque pixel de coordonnées *(x, y)*, l'intensité observée est le résultat d'un morphing. On peut écrire ce morphing sous la forme suivante :

$$I(x,y) = \alpha F(x,y) + (1-\alpha) B(x,y) = F'(x,y) + B'(x,y),$$

avec ***I*** qui est extrait d'une gaussienne de paramètres : $(\alpha \overline{\mu_1} + (1-\alpha)\overline{\mu_2} \quad , \alpha \overline{\overline{V_1}} + (1-\alpha)\overline{\overline{V_2}} \quad)$.

On peut donc écrire ce morphing sous la forme $X = \Theta + \mathrm{E}$, avec : $\begin{cases} loi \quad de \quad \Theta = G(\overline{\mu_\Theta}, \overline{\overline{V_\Theta}}) \\ loi \quad de \quad \mathrm{E} = G(\overline{\mu_\mathrm{E}}, \overline{\overline{V_\mathrm{E}}}) \end{cases}$.

Ce que l'on désire, c'est connaissant ***X***, quelle est la meilleure estimation de *Θ* que l'on puisse faire : cette réponse est donnée par les probabilités conditionnelles ; c'est : $E(\Theta | X)$.

Des calculs triviaux donnent alors la formule suivante :

$E(\Theta | X) = \mu_\Theta + \overline{\overline{M}}.(X - \mu_\Theta - \mu_\mathrm{E})$, avec $\overline{\overline{M}} = \overline{\overline{V_\Theta}}.(\overline{\overline{V_\Theta}} + \overline{\overline{V_\mathrm{E}}})^{-1}$.

Ainsi, connaissant la loi du morphing, on peut estimer l'image initiale cherchée (on calcule plutôt l'image *(R,G,B,α)* avec les coordonnées *R*, *G*, *B* représentant *α*-fois les véritables coordonnées de l'image initiale : cela permet d'éviter les erreurs d'approximation lorsque *α* est proche de *0*, et c'est une convention généralement adoptée pour stocker les valeurs d'une image).
On obtient donc finalement :

$$\alpha.\overline{color_1} = \alpha.\overline{\mu_1} + \overline{\overline{\alpha\, V_1}}\,.\,\overline{\overline{(\alpha\, V_1 + (1-\alpha)\, V_2)}}^{-1}.(\overline{X} - \alpha.\overline{\mu_1} - (1-\alpha)\overline{\mu_2})$$

, ce qui constitue l'estimation bayesienne de l'image originale.

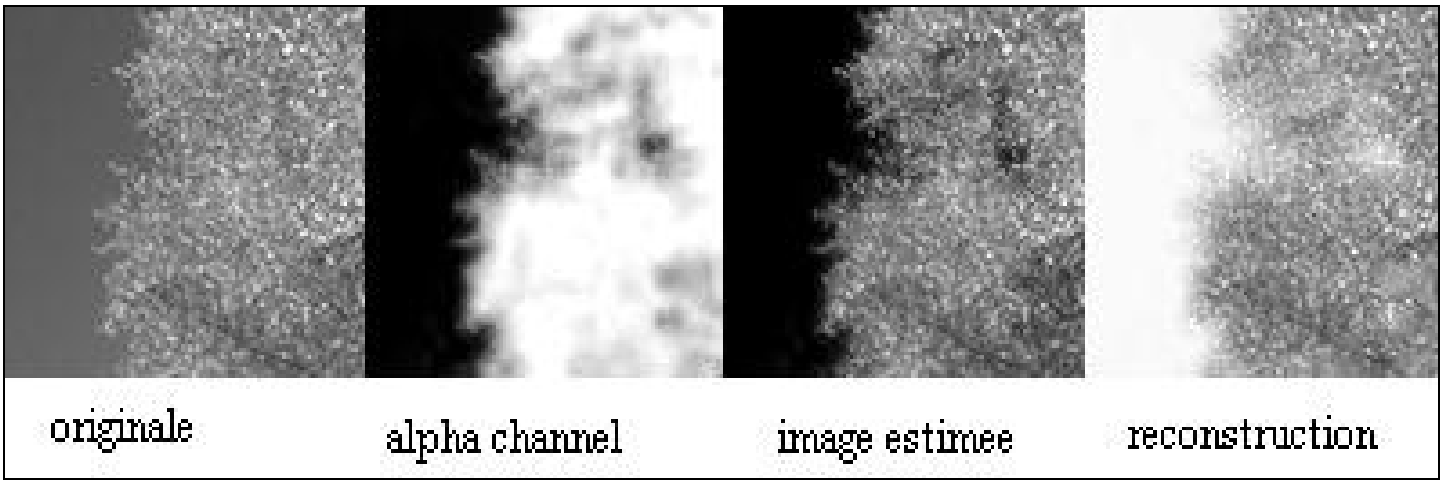
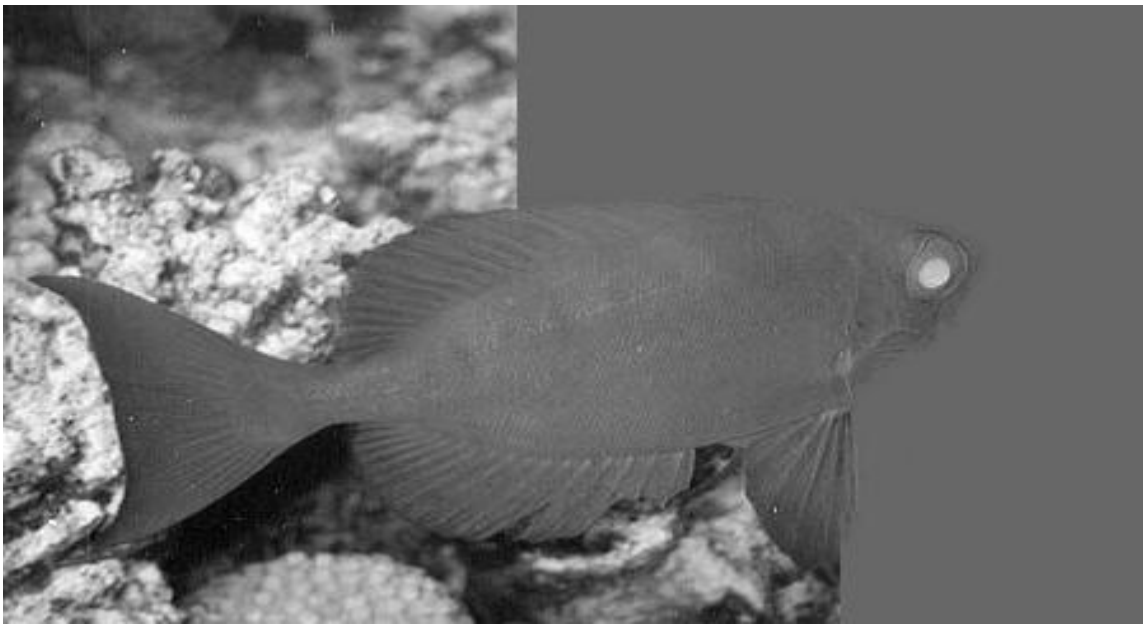
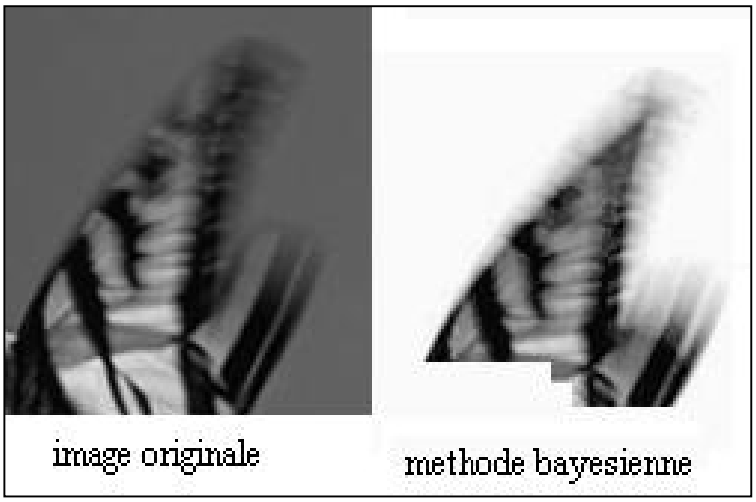


Figure 6: méthode bayesienne, reconstruction sur fond blanc (255,255,255)

Dans la troisième figure, l'estimation de l'image initiale affichée est l'*α*-image (cela explique donc le background noir !).





LIMITATIONS

1) *Temps de* calcul un peu lourd ; même si des améliorations certaines ont été trouvées, la construction des signatures est un peu lourde, et l'estimation en chaque pixel du coefficient alpha est particulièrement longue. Il est possible de pré-calculer les inverses des matrices définissant les ellipsoïdes le long de tous les segments du graphe, mais le temps d'estimation de chaque coefficient alpha reste un peu long.

- *Des situations conflictuelles* : ces dernières résultent souvent d'une approche globale, mais des conflits locaux peuvent toujours exister.

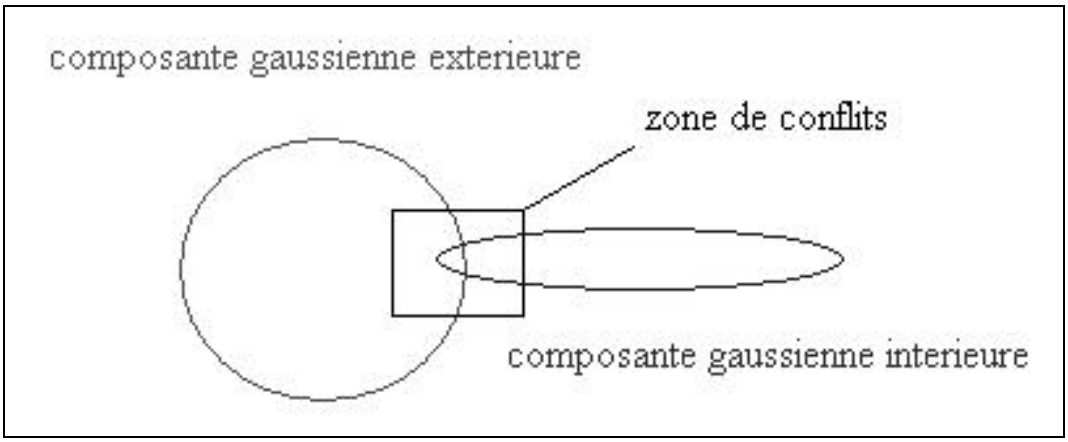
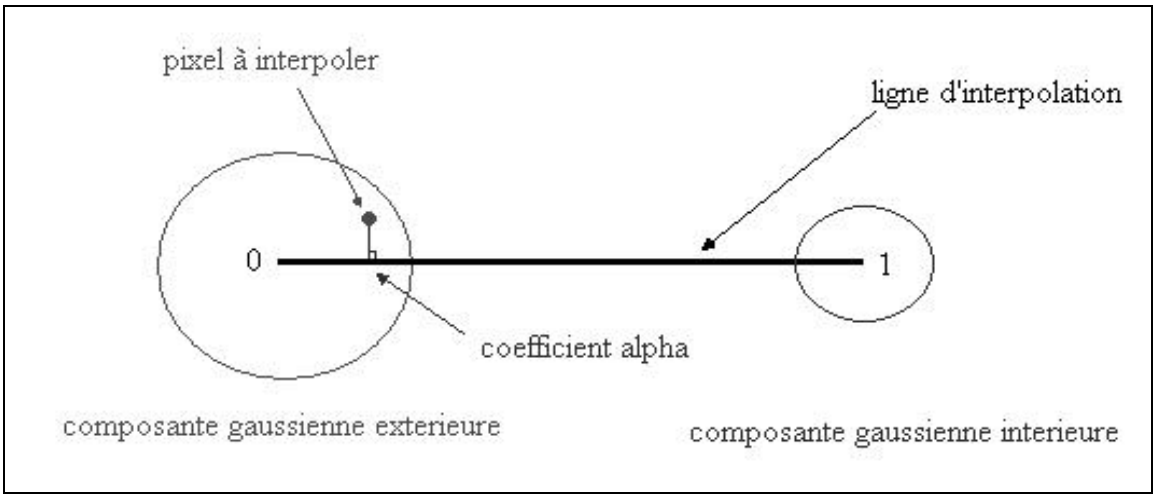


Figure 7: situation conflictuelle due au "bruit" de chaque composante

- *Une Précision quelquefois aléatoire* : dans certains cas, les coefficients alpha trouvés ne sont pas ceux qui conviennent et cela même si la fonction de probabilité possède un unique maximum bien défini. Le schéma ci-dessous montre un exemple simple pour lequel le coefficient alpha déterminé est différent de *0*, alors que le pixel considéré se situe dans la « zone de bruit » d'une composante. Bien-entendu des situations moins simples et beaucoup plus conflictuelles peuvent se produire.



Pour résoudre ce conflit, il suffit de définir un critère de sélection robuste. Ce critère peut être fondé sur la notion de distance de Mahalanobis, ou bien sur le résultat donné par les probabilités comme le montre la courbe ci-dessous.

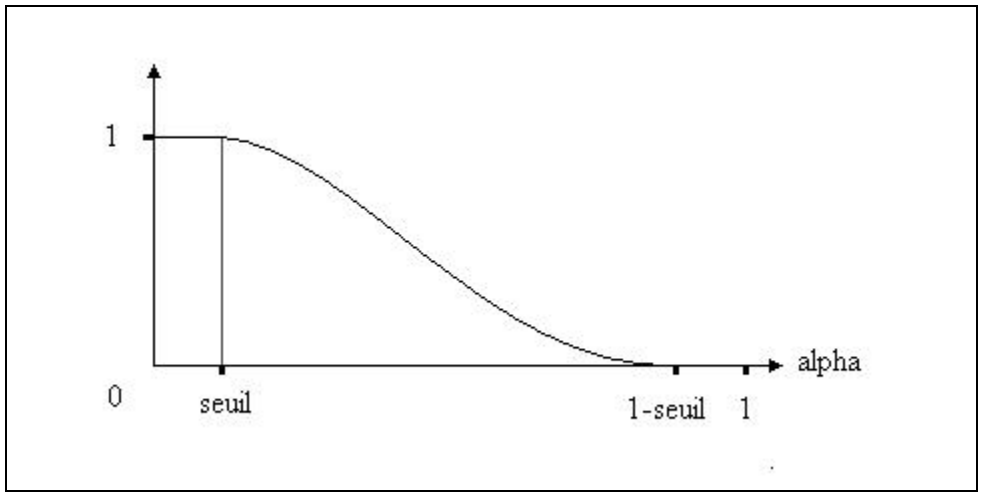
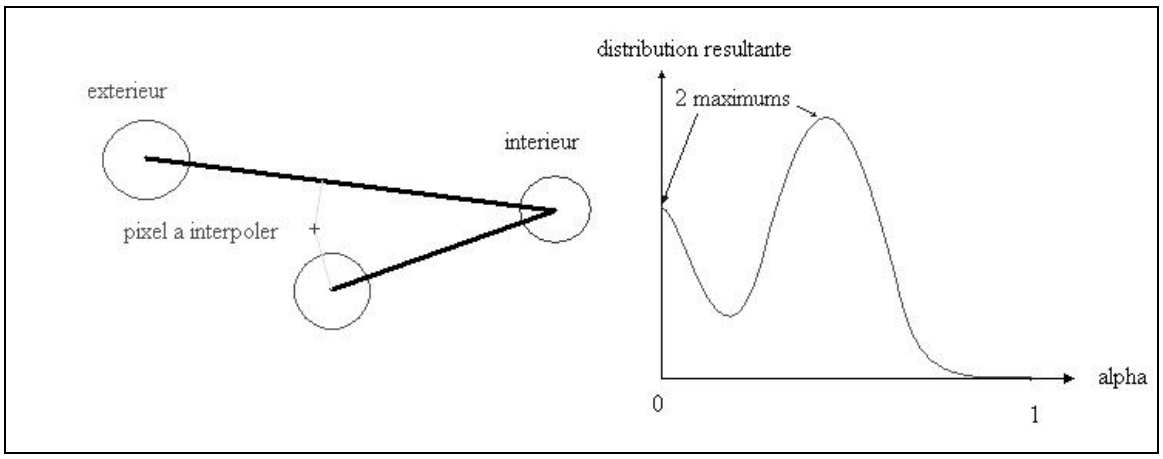


Figure 8:critère de sélection fondé sur le résultat du morphing

Dans d'autres cas il est possible que la courbe de distribution possède plusieurs maximums relatifs (un nombre inférieur au nombre de gaussiennes, c'est à dire le nombre de segments dans le graphe). Il est facile de détecter ce genre de conflits mais leur résolution n'est pas triviale. Une approche locale semble lever la plupart de ces problèmes, à l'exception de cas ou un travail sur la chromakey n'est pas adapté. Le schéma suivant montre l'un de ces problèmes.



Plusieurs pistes, autres que locales, pour résoudre ce genre de problème ont été explorées sans donner de résultats intéressants. La meilleure solution pour y pallier reste une observation des valeurs voisines, et un choix en conséquence.

Cette méthode pose quelques problèmes certains dont le plus délicat pourrait être le temps de calcul. En effet, la construction des deux signatures et de leur graphe correspondant n'est pas foncièrement lourde en calcul, mais une approche très locale, qui obligerait un calcul en chaque point, n'est pas concevable. De plus, l'estimation d'alpha en chaque pixel est coûteuse en temps de calcul, et il n'y a pas de moyen de contourner ce problème.

INTEGRATION

Plusieurs méthodes d'intégration sont possibles :

i. Développement d'un outil de retouche post-Rotoscopie.

ii. Une insertion directe dans le terme énergétique ne semble pas triviale. Cependant, semblable à l'approche en composantes globales de Nikos [7], il peut être possible de modifier l'énergie pour tenir compte globalement ou semi-localement des signatures.

b) Approche énergétique pour les snakes :

Le travail effectue par Nikos Paragios [7], synthétisé dans sa thèse intitulée « Geodesic Active Regions and Level Sets Methods », va être très utile dans l'élaboration d'une approche énergétique pour les snakes. Quelques limitations, notamment sur les temps de calculs nécessaires (pour la plupart, liés à l'approche Level Set), et la précision dans la segmentation, nécessitent de reprendre les méthodes abordées, mais ce travail est globalement très riche pour la diversité des approches ainsi que leur originalité.

Inclure un terme énergétique tenant compte de toute l'information de couleur n'est pas triviale : comme nous avons déjà pu l'évoquer, la couleur apparente d'un pixel est sensible à son illumination. Une approche intéressante est développée par Michele M. Covell & Trevor J. Darrell dans « dynamic occluding contours ». Dans cette publication, ils proposent un nouveau terme énergétique, qui tient justement compte de cette information couleur. Ce nouveau terme inclut une information de couleur locale, ainsi qu'un profil de variation autour de cette couleur. La grande difficulté de cette approche est de garder une correspondance entre points d'une image à l'autre, de manière à ce que l'information locale soit toujours valable. Cependant, même si cette approche est assez gourmande en temps de calcul, le terme « profil de variation » est relativement intéressant, et semble donner des résultats corrects.

Actuellement, un nouveau terme énergétique, tenant compte des travaux évoqués précédemment, est en cours d'élaboration.

c) Acquis certains :

Certains points nous paraissent évidents :

- Une approche paramétrée (courbe de Béziers, voire plus probablement B-Snakes, B-Splines : possibilité de définir des coins) semble nécessaire : facilité d'interaction utilisateur-tracé, temps de calcul « correct » (en comparaison avec les Level Sets, qui, bien que plus précises, sont bien plus gourmandes en calculs), méthode complètement adaptée a la rotoscopie, …

- L'interaction doit être optimale : la rotoscopie demande constamment de nombreuses retouches, qui doivent pouvoir être effectuées en « quasi-temps réel ». L'utilisateur ne doit pas être « rebuté » par la complexité des outils de retouche, qui doivent être évidents d'utilisation et

rapides (!) : l'**ergonomie**, la **facilite d'utilisation**, la **simplicité des outils**, sont des contraintes très fortes.

- Il est quasiment impossible de « rotoscoper » automatiquement une séquence d'images, mais les résultats fournis par une première passe de « segmentation-tracking » se doivent d'être les plus corrects possibles : la **robustesse** du logiciel est très importante. Pour ces raisons, une approche totalement déterministe ne devrait pas permettre d'obtenir des résultats toujours satisfaisants, et l'introduction de probabilités conditionnelles (voire peut-être même de « fuzzy logic », c'est à dire déférer certaines décisions à plus tard lorsque ces dernières ne sont pas triviales) semblent nécessaires. Notamment, au moins une « couche » de décisions déférées, afin d'éviter la prise en compte de possibles erreurs, devrait être utilisée.

- Suite aux remarques précédentes, il est obligatoire, afin de bien se positionner, de développer des **outils de correction efficaces**. La majorité des outils « Dutruc » devrait être incorporé dans le logiciel, et les travaux précédents devraient pouvoir être utilisés pour implémenter un outil de retouche post-Rotoscopie.

3) Résultats actuels et planning :

D'ici la fin du mois d'août, une API « garbage matte » sera implémentée, et quelques résultats du travail sur la chromakey insérés. Une première version 1.0 devrait être disponible fin novembre.

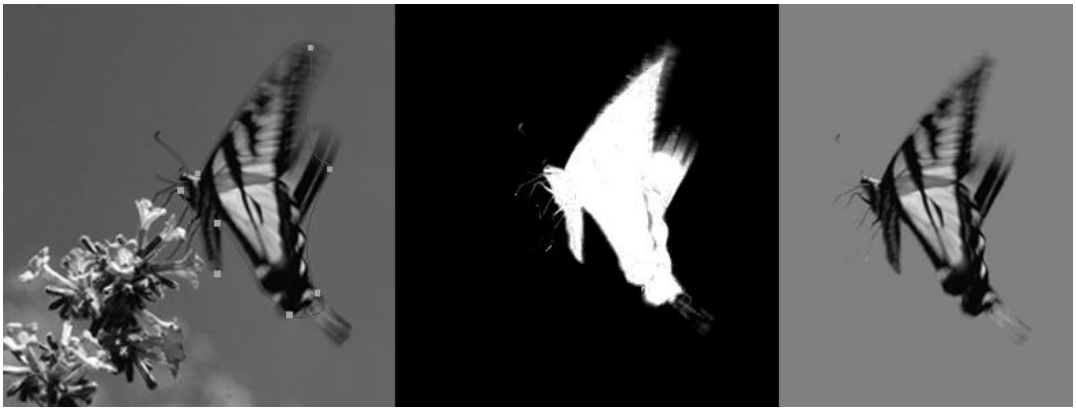


Figure 9: définition d' une courbe, calcul d' un mask et compositing suivant une méthode bayesienne

Figure 7 : outil de retouche post-retoscopie

Figure 8: intelligent scissors

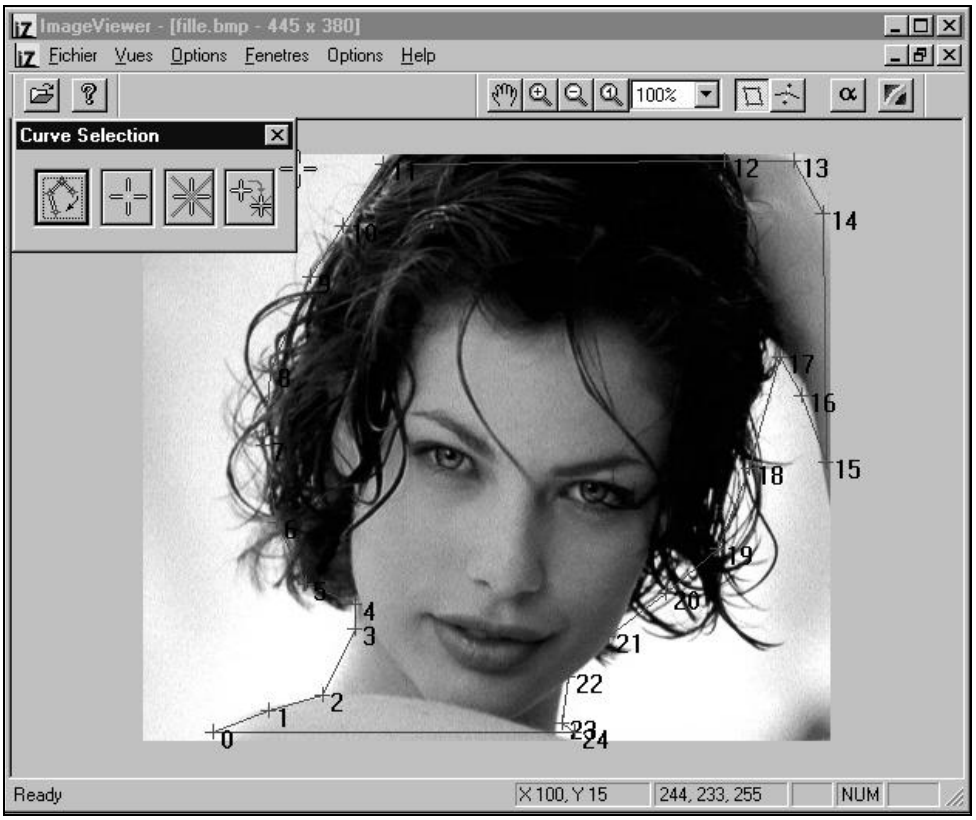


Figure 10: sélection d'une zone

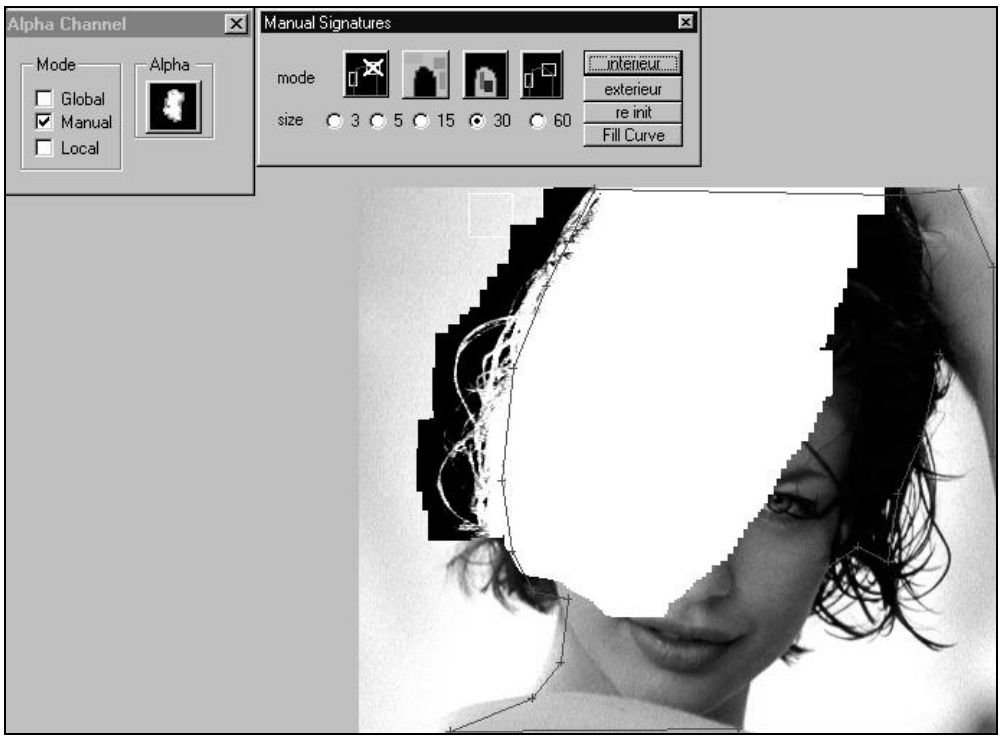


Figure 11: calcul de l'alpha channel

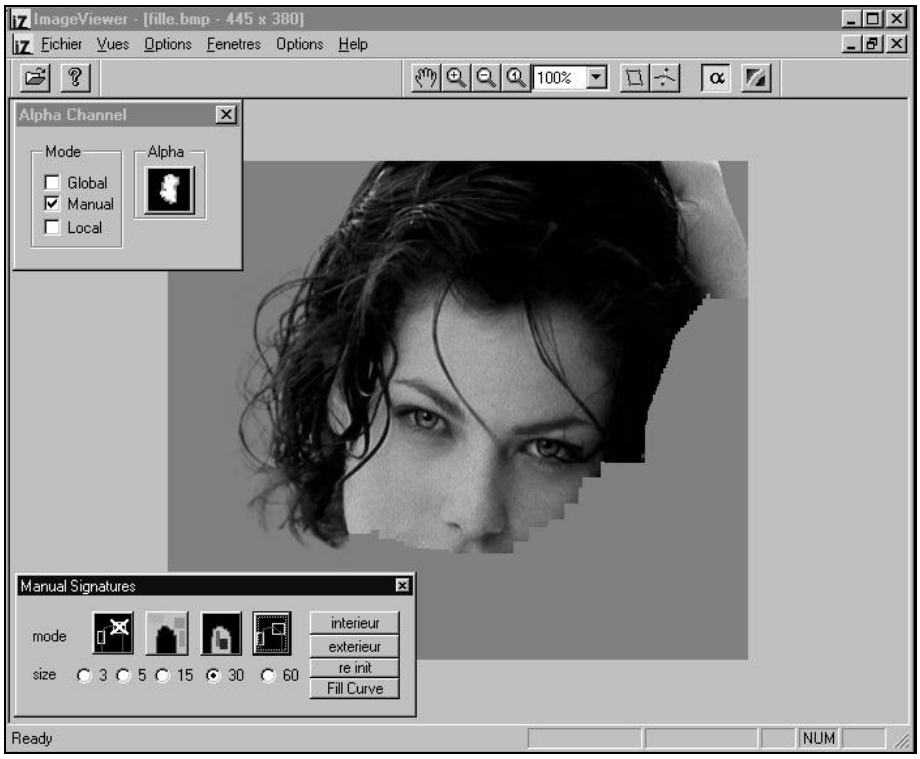


Figure 12: recompositing partiel

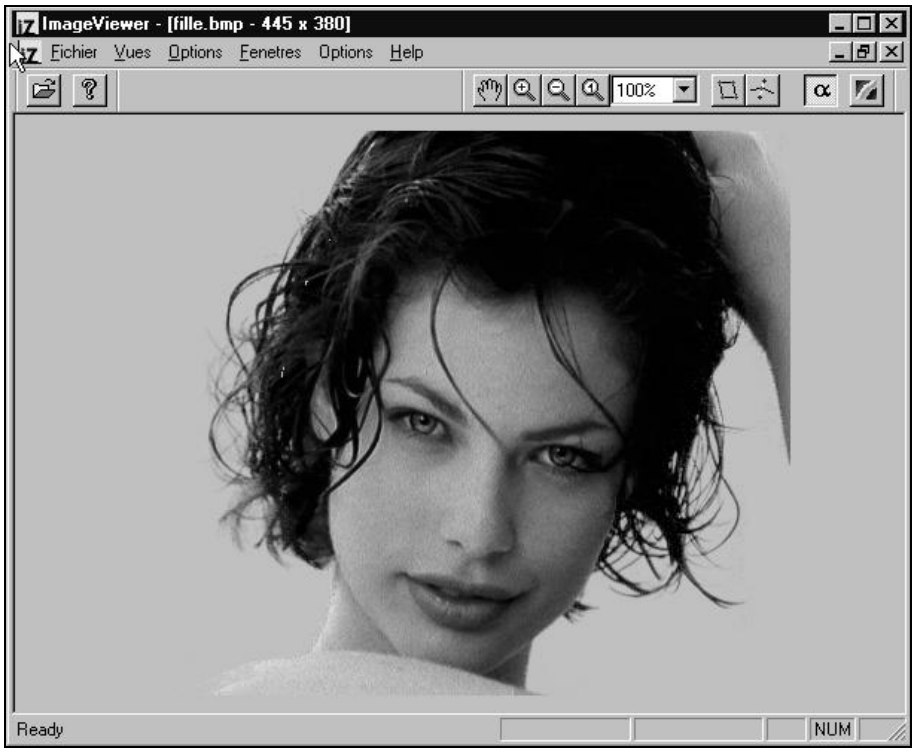


Figure 13: recompositing final

Figure 14: utilisation des B-Splines (information de gradient uniquement)

4) Références :

[1] Mark A.Ruzon & Carlo Tomasi
Alpha Estimation in Natural Images http://vision.stanford.edu/~ruzon/alpha/

[2] Michael Orchard & Charles Bouman
Color Quantization of Images IEEE

[3] Charles Poynton
Frequently Asked Questions about Color
Frequently Asked Questions about Gamma http://vera.inforamp.net/~poynton/

[4] David Bourgin
Color Spaces Frequently Asked Questions

[5] CIE Lab http://www.linocolor.com/colorman/sp_ciela_x.htm avec x={1,2,3,4}

[6] Dorin Comaniciu et Peter Meer
Distribution Free Decomposition of Multivariate Data

[7] Nikos Paragios & Rachid Deriche
PhD Thesis: Geodesic Active Regions and Level Sets Methods

[8] Tomoo Mitsunaga, Taku Yokoyama, Takashi Totsuka
Autokey: Juman Assisted Key Extraction

[9] Thomas Porter, Tom Duff
Compositing Digital Images

[10] Michele M. Covell & Trevor J. Darrell
Dynamic occluding contours

Annexe I : définition d'une composante gaussienne

Soient n pixels *i* dont $\vec{c}_i$ représente le vecteur couleur tridimensionnel dans l'espace approprie.

On définit l'espérance de cet ensemble de pixel par : $\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n}\vec{c}_i$, ainsi que sa matrice de covariance : $\underline{\underline{\text{cov}}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} {}^t\vec{c}_j \bullet \vec{c}_i - {}^t\vec{\mu} \bullet \vec{\mu}$.

Ceci étant fait on peut alors modéliser l'ensemble des *n* pixels par une composante gaussienne *G*. Nous définissons alors la 'probabilité' qu'un pixel de couleur $\vec{c}$ appartienne à la même composante couleur que les *n* pixels précédents. Elle est alors proportionnelle a :

$$G(\vec{c}, \vec{\mu}, \underline{\underline{\text{cov}}}) = \frac{1}{(2\pi)^{m/2}\sqrt{\det(\underline{\underline{\text{cov}}})}} \exp\left(-\frac{1}{2}{}^t(\vec{c}-\vec{\mu}) \bullet \underline{\underline{\text{cov}}}^{-1} \bullet (\vec{c}-\vec{\mu})\right)$$

(Ici m=3, car on travaille dans un espace tridimensionnel). On se rend alors compte que le facteur important est la distance entre la couleur $\vec{c}$ et l'espérance $\vec{\mu}$ au sens du produit scalaire définit par la forme bilinéaire : $\underline{\underline{\text{cov}}}^{-1}$.

Pour information seulement, la probabilité vraie qu'un pixel *i*, tiré au hasard dans l'ensemble, ait une couleur $\vec{c}$ vérifiant ${}^t(\vec{c}-\vec{\mu}) \bullet \underline{\underline{\text{cov}}}^{-1} \bullet (\vec{c}-\vec{\mu}) \leq a$ (c'est a dire que la couleur est incluse dans une ellipse E), est l'intégrale suivante : $\iiint_{\text{E}} G(\vec{x}, \vec{\mu}, \underline{\underline{\text{cov}}})\, d\vec{x}$.

Annexe II : définition de l'espace de couleur CIE Lab

L'espace classique de couleurs RGB est un espace qui se prête très mal au calcul mathématique dans le sens 'euclidien' du terme : notion de distance essentiellement. Cela signifie que deux points mathématiquement très proches peuvent être subjectivement très éloignés. Inversement, deux points paraissant subjectivement assez proches peuvent en fait être très éloignés au sens de leur distance euclidienne.

Pour cette raison, certains espaces de couleurs ont été crées, tels que les espaces CIE Lab et CIE Luv. Lors d'une première phase d'études, plusieurs espaces couleurs ont été testes : nous avons retenu l'espace couleur CIE Lab pour la qualité des résultats obtenus.

Pour une définition complète des différents espaces couleurs, nous renvoyons le lecteur curieux a [3], [4] et [5].

Tout d'abord définissons l'espace CIE XYZ, ou bien XYZitu (pour International Telecommunications Union):

RGB -> CIE XYZitu (D65)	CIE XYZitu (D65) -> RGB
*X = 0.431*Red+0.342*Green+0.178*Blue*	*Red = 3.063*X-1.393*Y-0.476*Z*
*Y = 0.222*Red+0.707*Green+0.071*Blue*	*Green = -0.969*X+1.876*Y+0.042*Z*
*Z = 0.020*Red+0.130*Green+0.939*Blue*	*Blue = 0.068*X-0.229*Y+1.069*Z*

L'espace de couleur CIE Lab a été introduit en 1976 par la CIE. Dans ce nouvel espace de couleur, L* est la luminance, a* et b* sont respectivement les chrominances rouge/bleu et jaune/bleu (On comprend alors pourquoi cet espace permet de réaliser des calculs plus intuitifs ; cf. sensibilité des bâtonnets…). Il est défini par rapport à l'espace CIE XYZ.

CIE XYZ -> CIE Lab

L=116((Y/Yn)^(1/3))-16* si *Y/Yn>0.008856*

ou

*L=903.3*Y/Yn* sinon.

a=500(f(X/Xn)-f(Y/Yn))*

b=200(f(Y/Yn)-f(Z/Zn))*

avec $f(t) = t^{1/3}$ si *Y/Yn>0.008856*

*f(t)=7.787 * t+16 / 116* sinon.

Avec ici (Xn,Yn,Zn)=(242,255,277). Voir [4] pour de plus amples informations.