

Contours actifs pour la stylisation cohérente de lignes animées

Pierre Bénard^{1,2}, Jingwan Lu³, Forrester Cole⁴, Adam Finkelstein³, Joëlle Thollot^{1,2}

¹Université de Grenoble, LJK

²INRIA

³Princeton University

⁴MIT CSAIL

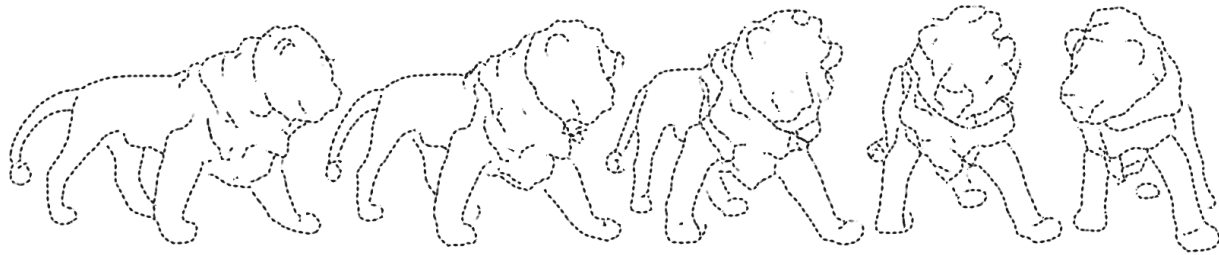


Figure 1: Dessin au trait d'un modèle 3D avec une stylisation cohérente au cours de l'animation.

Résumé

Cet article présente une méthode originale pour assurer la cohérence de la stylisation d'un dessin au trait extrait d'une scène 3D animée. Nous prenons en entrée des lignes dépendantes (silhouettes, contours suggestifs) ou non (crêtes, vallées) du point de vue. Ces lignes sont approximées et suivies par des polygones 2D tout au long de l'animation de telle sorte que leur paramétrisation reflète le mouvement des lignes représentées tout en restant uniforme par morceau. Cette paramétrisation peut alors être utilisée comme support pour une stylisation au travers d'une pyramide de texture. Notre approche est interactive et permet d'assurer la continuité temporelle d'une large gamme de styles.

Mots-clés : rendu non-photoréaliste, cohérence temporelle, contours actifs, dessin au trait

1. Introduction

Dans cet article nous nous intéressons à la stylisation d'un dessin au trait calculé à partir d'une scène 3D animée. Une fois les lignes d'intérêt extraites d'une scène 3D, le problème de leur stylisation vient de la difficulté à suivre ces lignes d'une image à l'autre et à leur assurer une paramétrisation cohérente dans le temps.

En particulier, les lignes dépendantes du point de vue, telles que les silhouettes, glissent à la surface des objets 3D. Elles sont par conséquent extraites indépendamment à chaque image et n'ont pas de lien naturel d'une image à l'autre. De plus, même en supposant que l'on dispose d'une méthode pour assurer le suivi de telles lignes, leur topologie et leur géométrie évoluent au fil de l'animation. C'est un problème particulièrement difficile dans le cas des silhouettes, celles-ci se situant à la frontière (*cusp*) de la visibilité. Or pour assurer une cohérence de leur stylisation en 2D il faut faire évoluer leur paramétrisation en accord avec leur mouvement 2D tout en la conservant uniforme dans la mesure du possible. Toute solution à ce type de problème sera nécessairement un compromis et selon la méthode utilisée créera

divers artefacts visuels tels que des glissements ou des changements brutaux.

Dans cet article nous proposons une approche qui permet d'assurer le suivi des lignes d'intérêt d'une image à l'autre en prenant en compte leur position en 2D mais aussi leur profondeur. Cela nous permet de distinguer des lignes qui seraient proches en 2D mais éloignées en 3D, comme c'est typiquement le cas pour des occlusions. Concrètement nous représentons les lignes d'intérêts par des polygones en 2D que nous nommons contours. Ces contours sont advectés, relaxés (sur le modèle des contours actifs) et nettoyés au fil de l'animation. La précision de ce suivi est définie par un rayon de couverture assurant que chaque ligne d'intérêt est représentée par un seul contour dans ce rayon.

D'autre part, nous proposons un mécanisme de propagation et de régularisation de la paramétrisation au cours du temps. En autorisant une ligne d'intérêt à être dessinée par plusieurs coups de pinceau † successifs, nous contrôlons les points de rupture de la paramétrisation. Un mécanisme de fusion permet d'éviter la multiplication de petits coups de pinceau au fil du temps.

†. Par coup de pinceau nous entendons un morceau de contour sur lequel sera plaquée une texture de style.

2. Travaux précédents

Stylisation cohérente de lignes De nombreux travaux ont porté sur l'extraction et le rendu de lignes à partir de modèles 3D (cf. [RCDF08] pour un panorama complet). Assurer la cohérence de la stylisation de ces lignes au cours de l'animation reste cependant un problème ouvert. En étendant les travaux de Masuch et al. [MSS98] et Bourdev [Bou98], Kalnins et al. [KDMF03] proposent le premier système complet de gestion de la cohérence des lignes stylisées pour des modèles de moyenne complexité. Un aspect central de leur méthode est la propagation de la paramétrisation des lignes d'une image à la suivante par le biais d'un buffer 2D. Cette approche pose deux problèmes principaux : (i) un risque d'aliasing lié à la résolution du buffer, (ii) le mélange de la paramétrisation des lignes proches en 2D après projection. Bénard et al. [BCGF10] suivent une approche comparable, en proposant une pondération visant à uniformiser progressivement les paramétrisations des lignes voisines. Nous sommes néanmoins convaincus que l'utilisation d'un buffer 2D constitue une limitation intrinsèque de ces approches, et nous proposons ici une méthode alternative de propagation de la paramétrisation basée sur le modèle des contours actifs (*snakes*) [KWT88].

Une seconde contribution de Kalnins et al. [KDMF03] est d'autoriser plusieurs coups de pinceau successifs par ligne d'intérêt. Leur paramétrisation est optimisée indépendamment en cherchant un compromis entre uniformité en espace écran et en espace objet. Un mécanisme permet d'uniformiser ces paramétrisations pour éviter la multiplication de petits coups de pinceau. Bénard et al. [BCGF10] évitent cette complexe optimisation en se limitant à un coup de pinceau par ligne, à des styles représentés en espace écran et en introduisant une nouvelle pyramide de textures auto-similaires – les *Self Similar Line Artmaps (SLAMs)* – qui résolvent en outre le problème du zoom infini. Nous nous restreindrons également à cette catégorie de styles pour utiliser les *SLAMs*, mais nous proposerons un modèle de coups de pinceau plus complexe.

Contours actifs en NPR Introduit par Kass et al. [KWT88], ce modèle de courbes dynamiques a été utilisé à plusieurs reprises en rendu non-photoréaliste. Dans le cadre du rendu peinture d'images et de vidéos, Hertzmann [Her01] représente les coups de pinceau sous la forme de contours actifs et adapte l'algorithme de relaxation par programmation dynamique d'Amini et al. [AWJ90] pour décider de leur placement.

Afin de segmenter et de réaliser le suivi des régions de couleurs dans des vidéos pour assurer la cohérence de leur stylisation, Agrawala et al. [Aga02, AHSS04] utilisent également ce modèle de contours appliqué à des splines de Bézier. L'utilisateur effectue manuellement le tracé initial des contours à suivre et ce dessin est propagé tout au long de l'animation. *SnakeToonz* [Aga02] alterne une phase de suivi des points de contrôle des splines avec une phase de re-

laxation gloutonne [WS92], tandis que dans [AHSS04] une optimisation non-linéaire globale des contours est effectuée sur l'ensemble de la séquence. Dans les deux cas, les performances ne sont pas interactives (plusieurs secondes par image) et l'utilisateur doit régulièrement corriger le résultat proposé par l'algorithme automatique, ce qui limite ces approches à de courtes séquences (une centaines d'images).

Couverture d'un dessin au trait Des solutions pour simplifier les dessins au trait ont été proposées dans le cas des images statiques aussi bien durant la stylisation [GDS04] qu'en post-traitement [BTS05, SC08]. Ces méthodes reposent sur la définition d'une couverture directionnelle, la première l'utilisant pour omettre certains traits ; les secondes pour remplacer un groupe de lignes par une ligne représentative. Cole et al. [CDF*06] proposent une solution dynamique basée sur un buffer de priorités calculé à partir de l'*item buffer* de Kalnins et al. [KMM*02]. Cette méthode contrôle la densité locale du dessin en omettant un certain nombre des lignes extraites (en modulant leur opacité). Elle ne permet cependant pas de modifier leurs positions pour représenter par une ligne "moyenne" deux lignes proches à l'écran.

En nous inspirant de [GDS04, BTS05], nous proposons une définition simplifiée de la couverture autorisant un contrôle dynamique. En outre, nous levons la limitation de [CDF*06] en rendant la position des coups de pinceau relativement indépendante des lignes extraites.

3. Suivi des lignes d'intérêt

Les lignes d'intérêt (silhouettes, crêtes, vallées, contours suggestifs, *apparent ridges* [RCDF08]) sont extraites à la surface d'un modèle 3D. Elles sont clippées dans le repère de la caméra et leur visibilité partielle est calculée en suivant la méthode de Cole et al. [CF10]. Nous utilisons leur structure de donnée – le *segment atlas* – pour les stocker sous forme de chemins discrétisés en 2.5D.

Notre objectif est de suivre ces lignes dans l'animation avec une tolérance contrôlée. Pour ce faire, nous définissons un ensemble de polygones 2D – les *contours* – se déplaçant en espace image. Ces contours doivent assurer une couverture unitaire, i.e. sans chevauchement, des lignes d'intérêt. Nous voulons de plus que la paramétrisation naturelle des contours reflète l'évolution de la ligne suivie en espace image et qu'elle soit explicitement propagée durant l'animation. Enfin, nous souhaitons maximiser la longueur des contours afin de laisser la plus grande liberté possible lors de la stylisation.

Notre algorithme est présenté Figure 2. Les contours sont advectés selon le champ de mouvement des lignes d'intérêt en considérant une transformation 3D rigide de ces lignes un entre deux images (Section 3.2). Cette transformation est exacte pour les lignes fixées sur la géométrie (vallées, crêtes) mais il s'agit d'une approximation pour les lignes dépendantes du point de vue (silhouettes en particulier) qui glissent à la surface du modèle. Pour compenser cette ap-

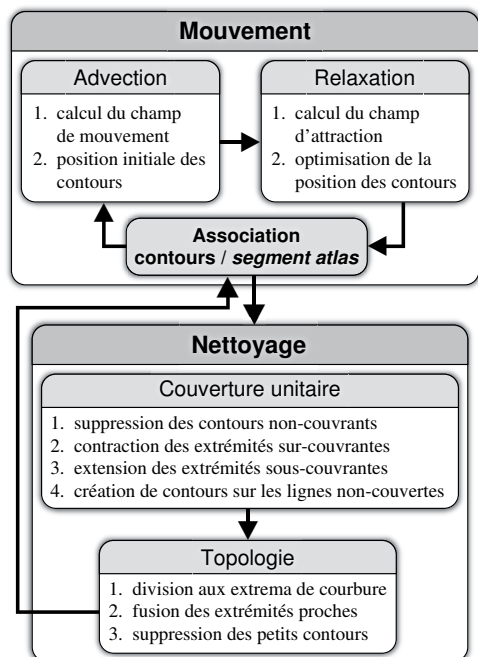


Figure 2: Algorithme de suivi des lignes d'intérêt, répété à chaque image de l'animation.

proximation, les contours sont relaxés en suivant le modèle des contours actifs (Section 3.3). Les contours sont enfin traités par des opérateurs topologiques (Section 3.4) afin de maximiser leur longueur.

3.1. Structure de données

Couverture Dans l'esprit de [GDS04, BTS05], nous définissons la couverture comme une fonction d'occupation dans un rayon r_{COUV} . Cette couverture n'est pas directionnelle afin d'accélérer les calculs. Les contours étant discrétisés, r_{COUV} doit être supérieur à la demie distance maximale entre deux sommets de la polyligne, Δs_{max} , pour fournir une approximation raisonnable de l'occupation réelle. Le pas de discrétisation des contours permet ainsi un compromis entre la finesse de la couverture et la complexité des contours.

Structure d'accélération Lors de l'advection et pour assurer une couverture unitaire, nous avons régulièrement besoin d'accéder aux voisins d'un sommet d'un contour ou d'un échantillon du *segment atlas* dans le rayon de couverture. Pour accélérer cette recherche nous utilisons la structure de données proposée dans [DME00]. Il s'agit d'une grille régulière creuse randomisée (implémentée par une table de hachage) contenant l'ensemble des sommets des contours ainsi que les échantillons du *segment atlas*. En choisissant $[2 \times r_{\text{COUV}}]$ comme taille de cellule, on peut énumérer tous les points dans un disque de rayon r_{COUV} en ne parcourant que quatre cellules.

3.2. Advection

Calcul du champ de mouvement Nous suivons une approche par reprojction similaire à celle utilisée par Bousseau et al. [BNTS07] et Lu et al. [LSF10], mais adaptée au mouvement des lignes. En effet, le champ de mouvement dense de la géométrie 3D ne nous est pas utile, seul celui des lignes extraites sur le modèle est significatif. Nous pouvons ainsi le calculer de façon exacte à partir du *segment atlas* et le stocker dans cette même structure.

Mise à jour des positions Pour trouver leur déplacement, les contours pourraient lire le champ de mouvement dans un buffer 2D, avec les mêmes limitations que décrites en Section 2 et, en particulier, une forte ambiguïté aux jonctions entre plusieurs lignes d'intérêt (premier et arrière-plan par exemple). Nous résolvons ce problème en conservant une profondeur par sommet du contour et en leur associant leur plus proche échantillon dans le *segment atlas* en 2.5D (Figure 3). Ainsi la cohérence du mouvement est assurée même si deux contours relativement proches en 2D suivent des lignes éloignées en 3D.

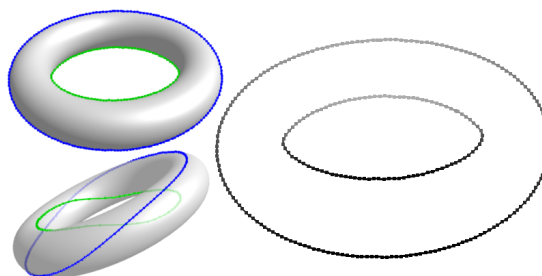


Figure 3: Profondeur par sommet des contours (droite) pour éviter les ambiguïtés 3D lors de l'advection.

3.3. Relaxation

Introduit par Kass et al. [KWT88] pour des applications en vision par ordinateur (segmentation, tracking, etc.), le modèle des contours actifs consiste en une courbe $\mathbf{v}(s) = (x(s), y(s))$, $s \in [0, 1]$ se déplaçant dans un domaine spatial pour minimiser l'énergie :

$$E = \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{ext}}(\mathbf{v}(s)) ds$$

L'énergie interne E_{int} assure la continuité (*tension* contrôlée par le paramètre α) et la régularité (*rigidité* pondérée par β) de la courbe :

$$E_{\text{int}}(\mathbf{v}(s)) = \frac{1}{2}(\alpha(s)|\mathbf{v}'(s)|^2 + \beta(s)|\mathbf{v}''(s)|^2)$$

L'énergie externe E_{ext} attire le contour près des zones d'intérêt de l'image. Dans notre cas, ces zones correspondent aux lignes d'intérêt du *segment atlas* en espace écran. Celles-ci sont donc projetées et rasterisées pour former une image binaire I . Cette image est filtrée par un noyau gaussien et son gradient est calculé pour obtenir le champ d'attraction :

$$E_{\text{ext}}(\mathbf{v}(s)) = -|\nabla [G_{\sigma}(\mathbf{v}(s)) \star I(\mathbf{v}(s))]|^2$$

La variance σ du filtre gaussien contrôle l'étendu du champ d'attraction de part et d'autre de la ligne d'intérêt. Plus σ est grand plus les contours peuvent être initialisés loin de leur position finale, mais plus les lignes d'intérêt interagissent entre-elles et donc moins le suivi est précis.

Nous suivons également l'approche originale de Kass et al. pour optimiser numériquement l'énergie E par descente de gradient. Le contour actif continu $\mathbf{v}(s)$ est discrétisé en n sommets v_i , $i \in \{0, \dots, n-1\}$ et l'énergie est minimisée itérativement par un schéma d'Euler semi-explicite. Pour chaque image, il ne requiert qu'une unique création et inversion de la matrice creuse (penta-diagonale) des forces internes, puis un certain nombre d'itérations en mettant à jour la position des sommets ainsi que les forces externes associées. Ce schéma est simple et relativement rapide (décomposition de Cholesky possible) à condition d'être correctement initialisé. Sous l'hypothèse d'un mouvement d'ampleur limité entre deux images, l'étape d'advection assure cette condition. Si cette hypothèse n'est pas vérifiée, il nous semble raisonnable de considérer que la cohérence du suivi ne serait de toute façon pas perceptible.

Pour permettre une croissance et décroissance importante du contour, en particulier durant un zoom, sa résolution n doit être mise à jour au bout de quelques itérations. Nous suivons l'approche de Delingette et al. [DME00] pour contrôler cette résolution et assurer une discrétisation relativement uniforme de la courbe. Ce ré-échantillonnage est particulièrement important dans la mesure où l'évolution de la paramétrisation est directement liée à celle des sommets du contour.

Le problème majeur des contours actifs est leur tendance intrinsèque à se contracter dans la direction tangente à la courbe en raison des énergies internes. Pour contrebalancer cet effet indésirable dans notre cas, nous considérons des forces masse-ressort entre chaque sommet de la courbe de telle sorte que celle-ci conserve une longueur quasi constante en l'absence de forces externes tangentielles. En pratique, cela consiste à ajouter un terme supplémentaire à l'énergie externe correspondant aux deux forces de rappel :

$$f_{i \rightarrow i+1} = k (L_i - \|v_{i+1} - v_i\|) \frac{v_{i+1} - v_i}{\|v_{i+1} - v_i\|}$$

$$f_{i \rightarrow i-1} = k (L_{i-1} - \|v_{i-1} - v_i\|) \frac{v_{i-1} - v_i}{\|v_{i-1} - v_i\|}$$

avec k la constante de raideur du ressort ($k = 1$ par défaut) et L_j la distance au repos du ressort correspondant à $\|v_{i+1} - v_i\|$ à l'image précédente.

3.4. Nettoyage

Création / initialisation Lorsqu'un nouveau contour est créé, et donc en particulier pour la première image de l'animation, celui-ci est initialisé avec les positions des échantillons du *segment atlas* non-couverts.

Suppression Les sommets d'un contour ne couvrant aucun échantillon du *segment atlas* après l'étape d'association sont supprimés. Cela peut induire la division d'un contour en plusieurs sous-parties et doit donc avoir lieu avant toute opération topologique.

Opérateurs topologiques Afin de remplir nos objectifs de précision, couverture et longueur des contours, nous définissons quatre opérateurs topologiques – contraction, extension, division et fusion (Figure 4) – dont les rayons d'action dérivent du rayon de couverture. Afin d'assurer une hystérésis suffisante (i.e. éviter une séquence instable de fusions/divisions ou extensions/contractions), ces rayons doivent suivre l'inégalité :

$$r_{\text{ext}} = r_{\text{fusion}} > r_{\text{couv}} > r_{\text{contract}} \geq \Delta s_{\text{max}}/2$$

et le cosinus de l'angle seuil θ_{max} , indiquant si deux contours sont localement parallèles, doit être pondéré par un facteur ϵ . En pratique nous utilisons $r_{\text{ext}} = 2 r_{\text{couv}} = 4 r_{\text{contract}} = 2\sqrt{2} \Delta s_{\text{max}}$, $\theta_{\text{max}} = \pi/6$ et $\epsilon = 0.1$

Ces opérateurs sont appliqués séquentiellement, de façon gloutonne, sur chaque sommet et/ou extrémité de chaque contour jusqu'à ce qu'ils ne produisent plus de modification. Ainsi, si l'ordre d'application des opérateurs est important, l'ordre des contours a un impact limité. Par exemple, deux contours en cours d'extension sur la même ligne d'intérêt vont croître alternativement d'un sommet et donc uniformément.

Chaque opérateur possède un critère limite c , ainsi qu'une fonction de coût f s'il peut être nécessaire de choisir le "meilleur" candidat parmi plusieurs dans son rayon d'action.

Contraction (Figure 4a)

$$c_{\text{contract}} = \begin{cases} |\vec{t}_i \cdot \vec{t}_j| & \geq \cos(\theta_{\text{max}}) + \epsilon \\ \|v_j - v_i\| & \leq r_{\text{contract}} \end{cases}$$

où le sommet v_j est un voisin de l'extrémité v_i et appartient à un contour différent, et \vec{t}_i (resp. \vec{t}_j) est la tangente au contour en v_i (resp. v_j).

Extension (Figure 4b)

$$c_{\text{ext}} = \begin{cases} \vec{t}_i \cdot \vec{t}_j & \geq \cos(\theta_{\text{max}}) + \epsilon \\ \|v_j - v_i\| & \leq r_{\text{ext}} \\ \neg \text{couvert}(v_j) \end{cases}$$

$$f_{\text{ext}} = \alpha \left(1 - \frac{\vec{t}_i \cdot \vec{t}_j}{\cos(\theta_{\text{max}}) + \epsilon} \right) + (1 - \alpha) \frac{\|v_j - v_i\|}{r_{\text{ext}}}$$

où v_i est l'extrémité d'un contour et \vec{t}_i sa tangente, v_j un échantillon du *segment atlas* et $\vec{t}_j = (v_j - v_i) / \|v_j - v_i\|$.

Division (Figure 4c)

$$c_{\text{div}} = \vec{t}_i \cdot \vec{t}_{i+1} \leq \cos(\theta_{\text{max}}) - \epsilon$$

où \vec{t}_i et \vec{t}_{i+1} sont les tangentes de deux sommets consécutifs d'un même contour.

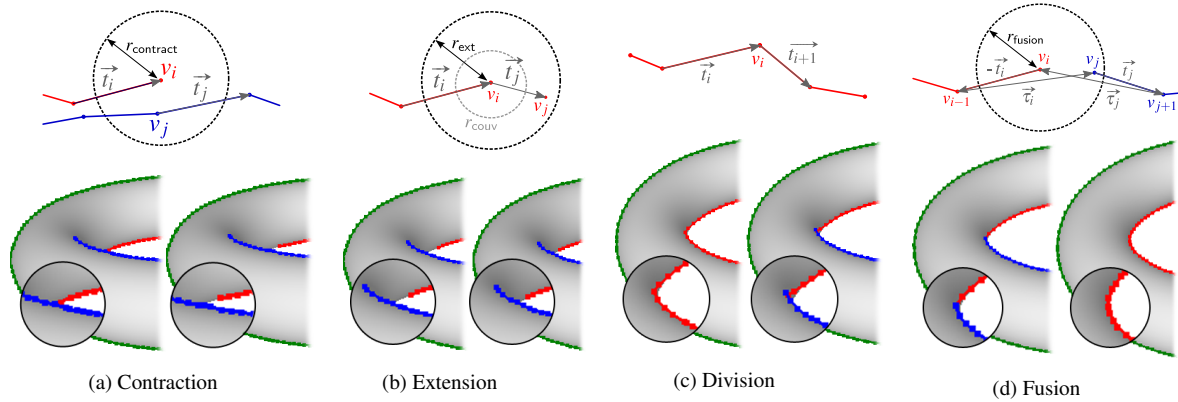


Figure 4: Opérateurs topologiques sur les contours.

Fusion (Figure 4d)

$$c_{\text{fusion}} = \begin{cases} \vec{t}_i \cdot \vec{t}_j \geq \cos(\theta_{\max}) + \varepsilon \\ \vec{t}_j \cdot -\vec{t}_i \geq \cos(\theta_{\max}) + \varepsilon \\ \|v_j - v_i\| \leq r_{\text{fusion}} \end{cases}$$

$$f_{\text{fusion}} = \alpha \left(1 - \frac{\min(\vec{t}_i \cdot \vec{t}_j, \vec{t}_j \cdot -\vec{t}_i)}{\cos(\theta_{\max}) + \varepsilon} \right) + (1 - \alpha) \frac{\|v_j - v_i\|}{r_{\text{fusion}}}$$

où v_i et v_j sont les extrémités de deux contours distincts, \vec{t}_i et \vec{t}_j leur tangent respective, $\vec{t}_i = (v_j - v_{i-1}) / \|v_j - v_{i-1}\|$ et $\vec{t}_j = (v_i - v_{j+1}) / \|v_i - v_{j+1}\|$.

Le facteur α est un facteur permettant la pondération entre le critère de proximité spatiale et directionnelle (par défaut $\alpha = 0.5$).

4. Stylisation temporellement cohérente

Les contours offrent un suivi temporellement cohérent des lignes d'intérêt, et donc un support pour propager une paramétrisation cohérente d'une image de l'animation à la suivante. Cependant, les événements topologiques discrets, en particulier les fusions, introduisent des discontinuités de paramétrisation indésirables. Pour y pallier, nous autorisons, comme Kalnins et al. [KDMF03], plusieurs coups de pinceau par contour (Figure 5b).

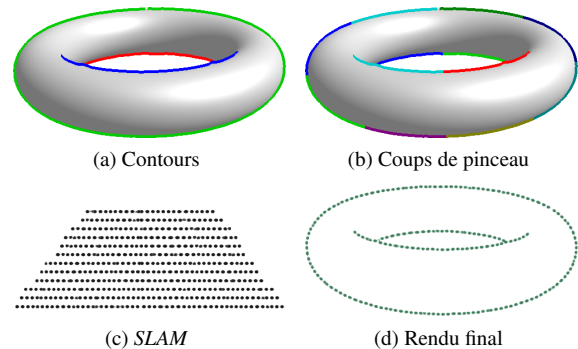
4.1. Coups de pinceau

Un coup de pinceau est une sous-partie d'un unique contour (chevauchements interdit) dont les sommets sont croissants (i.e. pas d'aller-retour le long du contour). Il s'en différencie fondamentalement par sa paramétrisation naturelle qui doit être linéaire et continue (Section 4.2).

La position d'un coup de pinceau est relative à celle du contour lui étant associé. Elle peut cependant en dévier pour répondre à d'autres contraintes de style. Ainsi, pour lisser la représentation par polygones, nous mettons à jours les positions des coups de pinceau en créant des splines cubiques

d'Hermite à partir d'un sous-échantillonnage des contours. Afin de maintenir la cohérence temporelle de ce lissage, les points de contrôle des splines sont propagés d'une image à la suivante (Figure 6). La fréquence d'échantillonnage (contrôlée par l'utilisateur) permet un compromis entre la fidélité à la ligne d'origine et un plus grand lissage des contours, sans pour autant impacter la qualité du suivi.

Le style du coup de pinceau est défini par l'utilisateur à travers quatre paramètres : une texture de médium auto-zoomable sous la forme d'une SLAM [BCGF10] (Figure 5c), une couleur, une épaisseur et une longueur maximale cible l_{\max} (potentiellement infinie).


 Figure 5: Coups de pinceau dont la longueur cible est $l_{\max} = 200$ pixels, et leur rendu final avec une SLAM de pointillés.

4.2. Paramétrisation

Pour que la texture de médium portée par le coup de pinceau soit uniforme en espace écran, sa paramétrisation naturelle doit suivre une relation linéaire : $T(s) = \rho s + \phi$, avec s son abscisse curviligne. Dans le même temps, la paramétrisation du coup de pinceau doit évoluer en suivant le mouvement et les changements topologiques des sommets du contour sous-jacent. Le principe de notre algorithme va ainsi consister à linéariser la paramétrisation à chaque image.

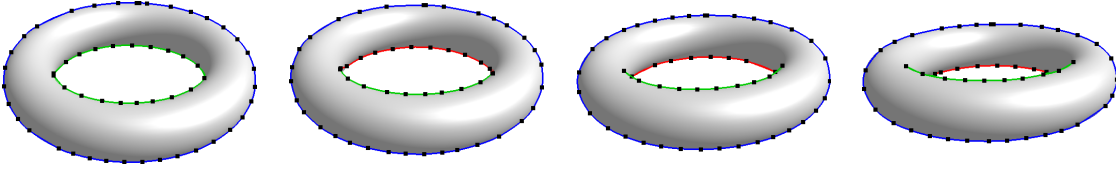


Figure 6: Lissage temporellement cohérent en utilisant des splines cubiques d'Hermite. Noter la propagation des points de contrôle (carrés noirs) y compris durant les changements topologiques.

Linéarisation A chaque image f de l'animation et pour chaque sommet i d'un coup de pinceau, nous traçons sa paramétrisation à l'image précédente t_i^{f-1} en fonction de son abscisse curviligne courante s_i et calculons les deux coefficients (ρ^f, ϕ^f) de la paramétrisation linéaire lui étant la plus proche au sens des moindres carrés. Contrairement à [BCGF10], l'utilisation d'un algorithme d'interpolation robuste aux données aberrantes mais coûteux comme *RANSAC* est inutile, les paramétrisations de deux coups de pinceau n'étant jamais mélangées. La paramétrisation de chaque sommet est alors mise à jour par $t_i^f = \rho^f s_i + \phi^f$.

Changements topologiques L'évolution d'un contour doit se traduire par celle de ses coups de pinceau, en particulier lors de son ré-échantillonnage et des changements topologiques. Ainsi, tous les coups de pinceau se divisent au même sommet que leur contour associé. Leurs extrémités se contractent ou s'étendent en même temps que celles du contour sous-jacent. Cependant, si deux contours fusionnent, leurs coups de pinceau ne fusionnent pas automatiquement. Dans le cas contraire, la continuité de leur paramétrisation naturelle linéaire serait brisée ce qui entraînerait une discontinuité temporelle.

Pour éviter ces artefacts, nous définissons un nouvel opérateur de fusion tenant compte de la paramétrisation. Deux coups de pinceau peuvent être fusionnés si :

- (i) ils sont adjacents sur leur contour associé,
- (ii) leur longueur cumulée $l_1 + l_2$ est inférieure à l_{\max} ,
- (iii) leur paramétrisation est compatible, i.e. :
 $|\rho_1 - \rho_2| < \mu$ et $|\phi_1 - \phi_2| < \mu$ (par défaut $\mu = 0.001$).

Auquel cas, les sommets des deux coups de pinceau sont mis en commun et leur paramétrisation devient (Figure 7) :

$$\tilde{T}(s) = \tilde{\rho}s + \tilde{\phi} \quad \text{avec} \quad \tilde{\rho} = (\tilde{\phi} - \phi_2)/l_1$$

Noter que seule la phase de T_2 est modifiée pour limiter le glissement local de la paramétrisation. La variation de fréquence est peu visible car elle correspond à un déplacement vertical global dans la *SLAM*.

Néanmoins, si le dernier critère a échoué mais que les deux paramétrisations sont suffisamment proches (en pratique $\mu = 0.05$), un mécanisme de nivellement est activé. Celui-ci consiste à faire converger progressivement la paramétrisation des deux coups de pinceau vers la paramétrisation commune attendue.

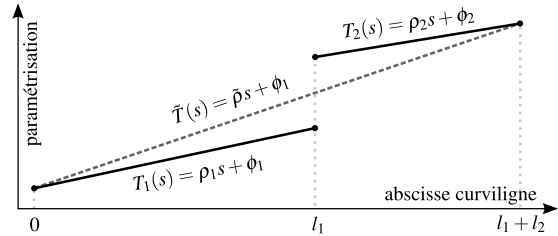


Figure 7: Processus de fusion de la paramétrisation de deux coups de pinceau.

5. Résultats

Comme l'illustre la figure 8 sur deux géométries complexes avec une longueur de coups de pinceau non bornée, notre approche permet de réduire considérablement le nombre de contours et d'augmenter leur longueur moyenne par rapport aux lignes d'intérêt directement extraites du modèle 3D et clippée en espace écran. Il en résulte, au moment du rendu, un dessin à la densité de lignes plus uniforme et avec de plus longs coups de pinceau. Aussi bien la méthode de Kalnins et al. [KDMF03] que celle de Bénard et al. [BCGF10] sont limitées par cette phase d'extraction et ne pourront pas créer de coups de pinceau plus longs que ces lignes d'intérêt. Noter cependant que dans le cas de modèles très lisses (comme le tore), notre approche ne présente pas de différence.

Les figures 8e,f montrent l'évolution au cours de l'animation du nombre et de la longueur des lignes d'intérêt, contours et coups de pinceau. On constate qu'aussi bien les contours que leurs coups de pinceau oscillent autour de leur valeur initiale quelque soit le nombre de lignes d'intérêt. Notre mécanisme de contrôle de la couverture semble donc bien fonctionner.

La variation de la longueur moyenne des contours traduit notre objectif de maximisation de leur longueur. On peut cependant constater que la longueur des coups de pinceau ne suit pas exactement la même tendance, mais redescend à une valeur plateau inférieure (néanmoins supérieure à celle des lignes d'intérêt) avant de s'y maintenir. Cela s'explique par le conservatisme des règles de fusion des coups de pinceau pour préserver leur paramétrisation et éviter les artefacts temporels.

Visuellement, la propagation de la paramétrisation semble bien se comporter, en particulier le long des lignes stables durant une séquence de l'animation (Figure 1, dos ou pattes

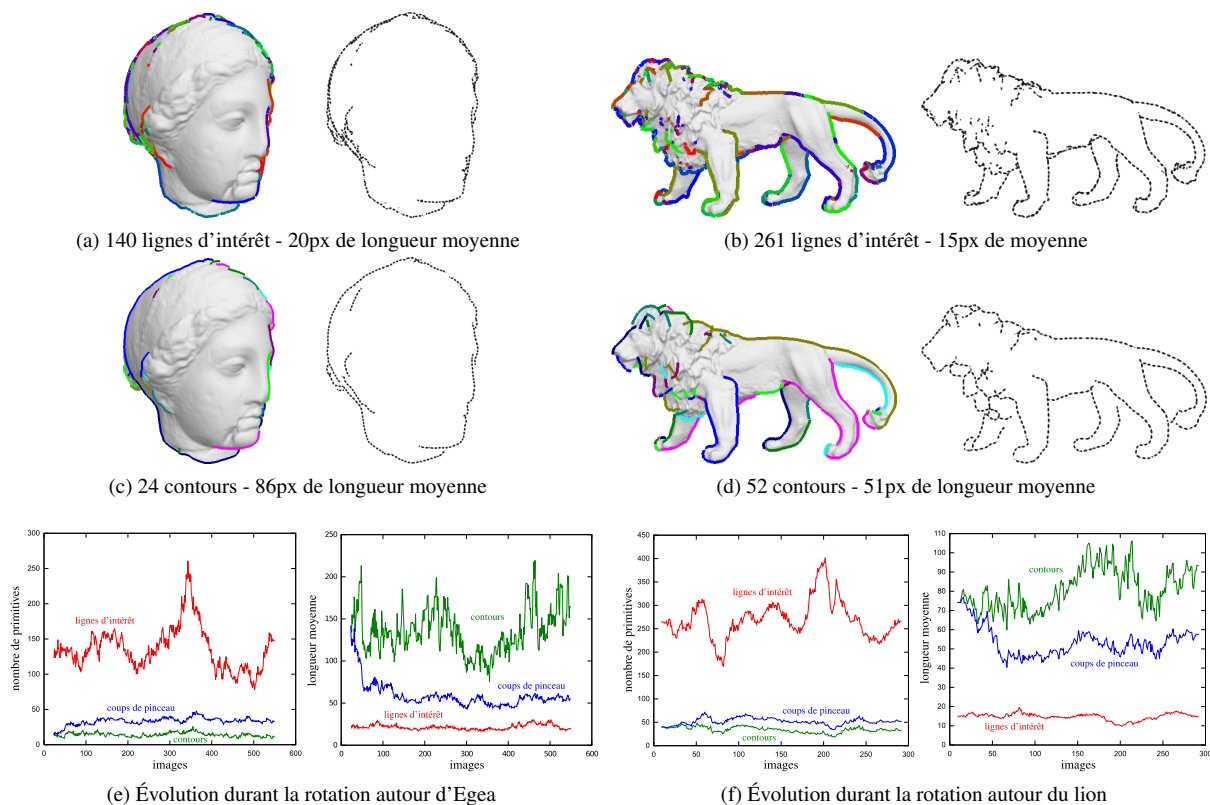


Figure 8: Application aux modèles d'Egea et du lion. Noter la réduction du nombre de contours en comparaison avec le nombre de lignes d'intérêt, et l'augmentation de leur longueur moyenne. Remarquer la simplification du rendu associé.

du lion par exemple). Bien que certaines lignes soient représentées par plusieurs coups de pinceau, et donc portent une paramétrisation linéaire par morceau, les variations de fréquences sont masquée par la *SLAM*. Cependant, des discontinuités liées aux différences de phase peuvent être plus ou moins visibles selon la texture utilisée.

Performances Notre implémentation essentiellement CPU – à l'exception du calcul de visibilité des lignes et du rendu final – atteint des performances interactives sur une machine récente (Intel Core 2 à 2.40 Ghz avec 4 GB de RAM et une GeForce GTX 260) pour des modèles de complexité raisonnable (Figure 9). Le principal goulot d'étranglement pour les modèles plus complexes est l'extraction des lignes d'intérêt qui est indépendante de notre approche. L'étape de relaxation est relativement peu coûteuse grâce à l'utilisation de CHOLMOD [CDHR08] pour inverser efficacement la matrice creuse, et ce bien que nous utilisons un nombre fixe d'itérations. Les transferts CPU/GPU pour calculer le champ d'attraction représentent un coût non négligeable qu'il devrait être possible d'optimiser.

6. Discussion et travaux futurs

L'approche que nous avons présentée permet l'évolution cohérente de la paramétrisation de lignes d'intérêt. Notre ap-

	9.6k tris	270k tris
Images par seconde	12	6
Extraction des lignes	34%	44%
Champ d'attraction	25%	18%
Relaxation	7%	3%
Opérations topologiques	5%	3%

Figure 9: Performances pour un modèle simple et un modèle complexe (taille d'image de sortie de 512x512 pixels).

proche est robuste grâce au fait que nous n'imposons qu'une paramétrisation linéaire par morceau à travers la séparation entre contours et coups de pinceau. Les mécanismes de nettoyage mis en oeuvre permettent d'éviter une dégénérescence au fil de l'animation.

Nous avons pour l'instant utilisé des paramètres par défaut pour nos expérimentations. Une étude plus approfondie et systématique de ces paramètres est nécessaire pour aller plus loin. En particulier, il n'est pas évident de savoir quels paramètres sont utiles à l'utilisateur et dans quels intervalles de valeurs. Cependant, en les sélectionnant habilement, ils pourraient permettre l'introduction automatique de variations dans l'esprit de [TJ07].

Les stylisations que nous avons expérimentées sont assez simples et, dans le futur, nous souhaitons étendre la gamme de styles possibles en s'appuyant sur notre couche de contours cohérents. Plusieurs pistes sont possibles en étendant notre modèle de coup de pinceau. On peut bien-sûr ajouter des effets standards tels qu'un masque de transparence ou de l'*overshoot* pour déborder des lignes d'intérêt. Mais on peut également envisager des abstraction géométriques plus drastiques qu'une simple spline comme des clothoïdes ou des segments de droites. Enfin, nous voudrions aussi permettre le chevauchement des coups de pinceau ce qui implique d'étendre notre notion de couverture pour gérer une densité plus complexe.

Notre approche est suffisamment générale pour s'appliquer à des vidéos en utilisant des algorithmes de vision pour extraire une approximation des données dont nous avons besoin : champ de mouvement, lignes d'intérêt, profondeur.

References

- [Aga02] AGARWALA A. : Snakatoonz : a semi-automatic approach to creating cel animation from video. In *NPAR '02 : Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002). 2
- [AHSS04] AGARWALA A., HERTZMANN A., SALESIN D. H., SEITZ S. M. : Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH '04 : ACM SIGGRAPH 2004 Papers* (2004). 2
- [AWJ90] AMINI A. A., WEYMOUTH T. E., JAIN R. C. : Using dynamic programming for solving variational problems in vision. *IEEE Trans. Pattern Anal. Mach. Intell.* (1990). 2
- [BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A. : Self-Similar Texture for Coherent Line Stylization. In *NPAR 2010 : Proceedings of the 8th International Symposium on Non-photorealistic Animation and Rendering* (2010). 2, 5, 6
- [BNTS07] BOUSSEAU A., NEYRET F., THOLLOT J., SALESIN D. : Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics* (2007). 3
- [Bou98] BOURDEV L. : *Rendering Nonphotorealistic Strokes with Temporal and Arc-Length Coherence*. Master's thesis, Brown University, 1998. 2
- [BTS05] BARLA P., THOLLOT J., SILLION F. : Geometric clustering for line drawing simplification. In *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering2005)* (2005). 2, 3
- [CDF*06] COLE F., DECARLO D., FINKELSTEIN A., KIN K., MORLEY K., SANTELLA A. : Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering* (2006). 2
- [CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S. : Algorithm 887 : Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 3 (2008). 7
- [CF10] COLE F., FINKELSTEIN A. : Two Fast Methods for High-Quality Line Visibility. *IEEE Transactions on Visualization and Computer Graphics* (2010). 2
- [DME00] DELINGETTE H., MONTAGNAT J., EPIDAURE P. : New algorithms for controlling active contours shape and topology. In *European Conference on Computer Vision* (2000), pp. 381–395. 3, 4
- [GDS04] GRABLI S., DURAND F., SILLION F. X. : Density measure for line-drawing simplification. In *PG '04 : Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (2004). 2, 3
- [Her01] HERTZMANN A. : Paint by relaxation. In *Computer Graphics International 2001* (2001). 2
- [KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A. : Coherent stylized silhouettes. *ACM Transactions on Graphics* (2003). 2, 5, 6
- [KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A. : WY-SIWYG NPR : drawing strokes directly on 3d models. In *ACM Transactions on Graphics* (2002). 2
- [KWT88] KASS M., WITKIN A., TERZOPOULOS D. : Snakes : Active contour models. *International Journal of Computer Vision* (1988). 2, 3
- [LSF10] LU J., SANDER P. V., FINKELSTEIN A. : Interactive painterly stylization of images, videos and 3d animations. In *I3D '10 : Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010). 3
- [MSS98] MASUCH M., SCHUMANN L., SCHLECHTWEG S. : Animating Frame-to-Frame Consistent Line Drawings for Illustrative Purposes. In *SimVis* (1998). 2
- [RCDF08] RUSINKIEWICZ S., COLE F., DECARLO D., FINKELSTEIN A. : Annotated bibliography : Published research on line drawings from 3D data. In *SIGGRAPH 2008 Course Notes : Line Drawings from 3D Models* (2008). <http://tinyurl.com/S08LinesCourse>. 2
- [SC08] SHESH A., CHEN B. : Efficient and dynamic simplification of line drawings abstract. *Computer Graphics Forum* (2008), 537–545. 2
- [TJ07] TWIGG C. D., JAMES D. L. : Many-worlds browsing for control of multibody dynamics. *ACM Transaction on Graphics* (2007). 7
- [WS92] WILLIAMS D. J., SHAH M. : A fast algorithm for active contours and curvature estimation. *CVGIP : Image Underst.* (1992). 2