Un algorithme semi-guidé performant de colorisation en aplats pour le dessin au trait

Sébastien Fourey, David Tschumperlé, David Revoy

Un algorithme semi-guidé performant de colorisation en aplats pour le dessin au trait

Sébastien FOUREY[1], David TSCHUMPERLÉ[1], David REVOY[2]

[1]Laboratoire GREYC (CNRS UMR 6072), Equipe Image,
6 Bd Maréchal Juin, 14050 Caen Cedex

[2]Artiste indépendant, Montauban, France

Sebastien.Fourey@ensicaen.fr, David.Tschumperle@ensicaen.fr, http://www.davidrevoy.com

Résumé – Nous présentons un algorithme rapide et efficace de colorisation semi-automatique de dessins au trait (de type bande dessinée), basé sur deux étapes principales: 1. L'analyse de la géométrie locale des traits de dessin et leur fermeture éventuelle par des *splines*/segments, et 2. la colorisation rapide par le remplissage de régions connexes avec des couleurs aléatoires, ou par l'extrapolation de marqueurs colorés placés par l'utilisateur. Notre méthode génère des colorisations en aplats de qualité équivalente aux méthodes de l'état de l'art, tout en proposant une complexité algorithmique très réduite, ce qui se traduit par une interactivité accrue de l'outil pour l'utilisateur.

Abstract – We present a fast and efficient algorithm for the semi-supervised colorization of line-arts (e.g. hand-made cartoons), based on two successive steps: 1. The analysis of the local stroke geometry, and their possible closing by *splines*/segments, and 2. The colorization by the filling of connected regions with random colors, or by the extrapolation of color markers set by the user. Our method renders colorization of similar quality compared to state of the arts algorithms, while having a lower algorithmic complexity, leading to more possible user interactivity.

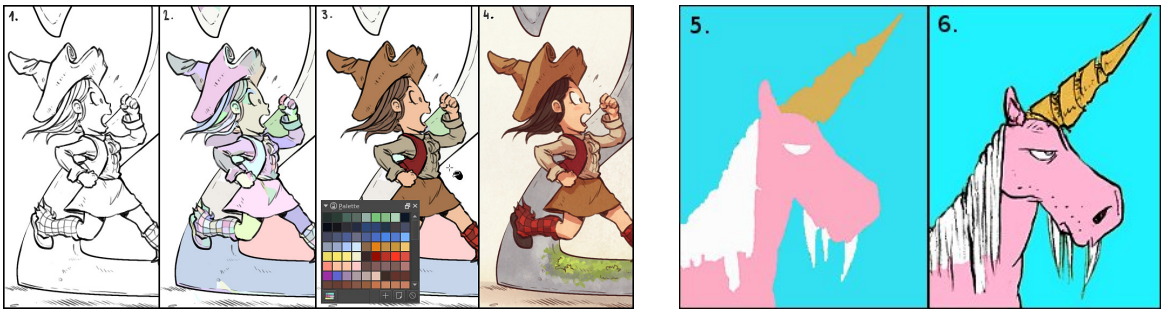


FIG. 1: Rôle et technique de la colorisation de dessins *en aplats* pour la bande dessinée.

1 Introduction

Dans le domaine de la bande dessinées, la colorisation d'un dessin au trait (aussi appelé *line-art*, Fig. 1.1) se réalise en deux étapes successives : Le dessin est d'abord pré-colorisé par *aplats* (Fig. 1.3), c.-à-d. en attribuant une couleur unique à chaque région ou objet distinct du dessin d'origine. La personne préposée à ce travail spécifique de pré-coloriage est nommé *préparateur d'aplats*. Dans un second temps, le *coloriste* retravaille ce pré-coloriage, ajoutant ombres, lumières et ambiance colorimétrique, pour obtenir le résultat de colorisation final (Fig. 1.4). Pratiquement, la colorisation en aplats aboutit à la création d'un nouveau calque qui ne contient que des zones de couleurs constantes par morceaux, formant ainsi une partition colorisée du plan (Fig. 1.5). Ce calque est fusionné avec le *line-art* d'origine pour un rendu du dessin colorisé en aplats (Fig. 1.6).

Coloriste et préparateur d'aplats sont généralement deux personnes différentes. Les artistes l'avouent eux-même : la colorisation en aplats est un processus long et fastidieux, nécessitant patience et précision. Les outils « classiques » présents dans les logiciels de peinture numérique ou de retouche d'images ne rendent en effet pas cette tâche aisée : Par exemple, les outils de remplissage par croissance de région, même les plus évolués, gèrent très mal les discontinuités dans les traits de dessin ainsi que les traits *anti-aliasés* (Fig. 2a). Il est alors courant que l'artiste effectue ses aplats en peignant manuellement ses couleurs avec une brosse sur un calque séparé, avec tous les problèmes de précision (notamment aux abords des traits) que cela suppose (Fig. 2b,c). Il peut arriver que l'artiste décide de contraindre explicitement son style de dessin, en utilisant par exemple des lignes crenelées en résolution supérieure (plutôt que des lignes *anti-aliasées*), et/ou en ne se forçant

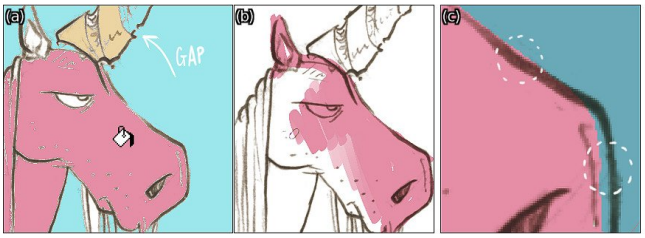


FIG. 2: Problèmes rencontrés pour la création d'aplats via les outils classiques des logiciels de dessins/peintures numériques.

à tracer des traits sans « trous », pour faciliter le travail de colorisation du préparateur d'aplats. Récemment, quelques travaux de recherche ont abordé ces problèmes, en proposant des algorithmes dédiés à la colorisation en aplats, d'abord en photographie [1], puis plus spécifiquement pour des dessins aux traits [2, 3, 4]. Ces approches se basent toutes sur des étapes similaires : l'utilisateur doit placer des marqueurs colorés aux endroits clés de son dessin. Ces marqueurs sont ensuite extrapolés pour tous les pixels de l'image, par la *minimisation d'une énergie globale* prenant en compte la géométrie des traits du dessin. Malheureusement, la complexité algorithmique de ces processus de minimisation itératifs est importante, et l'interactivité pour l'utilisateur s'en trouve très diminuée, notamment pour la colorisation d'images de grande résolution (planches A4 entières de BD par exemple, de résolution typique *5000×7000*).

Dans cet article, nous proposons une technique originale de colorisation en aplats, donnant des résultats de qualité similaires à ces techniques d'optimisation, mais avec une complexité algorithmique bien moindre, permettant une meilleure interactivité et une utilisation confortable pour l'utilisateur même sur des machines modestes. L'algorithme proposé est capable de bien gérer les dessins au trait contenant des discontinuités de tracé, ainsi que les traits *anti-aliasés*. Il se base principalement sur une analyse fine de la géométrie locale (normales et courbures) des traits de dessin, suivi d'une étape de fermeture de régions par des courbes *splines* ou des segments. Il permet enfin de choisir deux modes de colorisation en aplats : aléatoire (Fig. 1.2) ou guidés par des marqueurs de couleur placés par l'utilisateur.

2 Fermeture de traits binarisés

On considère une image de dessin au trait $I : \Omega \to [0, 255]$ (de taille $w \times h$) que l'on souhaite coloriser. Il s'agit d'une image en niveaux de gris définie sur le domaine discret $\Omega = \{0, \ldots, w-1\} \times \{0, \ldots, h-1\}$, avec des valeurs réelles puisque le dessin est réalisé à priori avec des traits *anti-aliasés* (Fig. 1.1). L'algorithme que nous proposons s'appuie sur une étape cruciale de fermeture des traits de dessin, qui se réalise sur une version binarisée de I, dénotée $I_b : \Omega \to \{0, 1\}$.

2.1 Pré-traitement de l'image anti-aliasée

Binarisation de l'image anti-aliasée : Pour $\theta \in [0, 255]$, on définit l'image binarisée I_b par simple seuillage de I, comme suit :

$$\begin{array}{rccl} I_b : & \Omega & \longrightarrow & \{0,1\} \\ & (x,y) & \longmapsto & \begin{cases} 1 \text{ si } I_{(x,y)} \geq \theta, \\ 0 \text{ sinon.} \end{cases} \end{array} \quad (1)$$

Cette étape permet de simplifier le problème à celui de la fermeture de traits d'une image binaire. Le fait que cette simplification intervienne sans perte de généralité ni d'efficacité de la méthode proposée est discuté dans la section 4.

Estimation de la largeur des traits : Afin de rendre la méthode de fermeture indépendante de la résolution de l'image, une étape préliminaire permet de réduire si besoin l'épaisseur des tracés à quelques pixels, en utilisant une érosion morphologique. Le rayon utilisé pour cette érosion est déterminé automatiquement par estimation de la largeur des traits présents dans le dessin. Une transformée en distance euclidienne [5] permet dans un premier temps de calculer la distance maximale au fond de chacune des composantes 8-connexes de pixels de trait ; la valeur médiane de tous ces maxima fournit une bonne approximation de la demi-épaisseur des traits. Il est à noter que d'éventuelles déconnexions provoquées par l'érosion appliquée, qui restent rares, seront de toute façon compensées par l'étape suivante de fermeture des traits.

2.2 Caractérisation des points d'intérêt

La suite de la méthode repose sur la détection de point d'intérêts qui sont situés aux extrémités des traits de dessin, et par la même occasion sur l'estimation d'une direction convenable pour éventuellement prolonger ces traits (Fig. 5c). Afin de caractériser ces points d'intérêt, nous proposons d'estimer le champ des normales et courbures autour des zones non nulles de I_b par une méthode inspirée de [6]. Cette méthode, qui repose sur un moyennage géométrique local, est décrite ci-après.

2.2.1 Estimation des normales et courbures aux bords

Avant de présenter la méthode d'estimation des normales au bord d'une région (vecteurs verts de la Fig. 5b), nous commençons par définir précisément la notion de contour inter-pixels. Nous suivons en cela l'approche correspondant aux *bi-curves* de [7, Section 4]. On désigne par $X \subset \Omega$ l'ensemble des pixels de valeur non-nulle de I_b, et on note $\overline{X} = \Omega \setminus X$. Classiquement, deux pixels $p = (x_p, y_p)$ et $q = (x_q, y_q)$ sont dits 4-adjacents si $|x_p - x_q| + |y_p - y_q| = 1$, auquel cas ils partagent une arête, aussi appelée *lignel*. Le *bord* de X, noté $\delta(X)$, est l'ensemble des arêtes partagées par deux pixels 4-adjacents dont l'un est dans X et l'autre dans $\overline{X}$. Une arête pouvant être identifiée par un couple de pixels, on a $\delta(X) = \{(p, q) \in X \times \overline{X}, p \text{ et } q \text{ sont 4-adjacents}\}$. Sauf précision contraire, dans la suite les *lignels* (ou arêtes) sont supposés appartenir au bord de X.

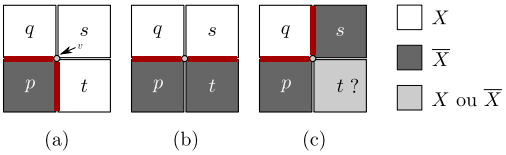


FIG. 3: Relation d'adjacence entre *lignels*. Étant donné 4 pixels p, q, s et t tels que $(p,q) \in \delta(X)$, un seul *lignel* incident au sommet v est en relation avec (p,q) par $\mathscr{R}_a$. Ainsi : $(p,q)\mathscr{R}_a(p,t)$ dans le cas (a), $(p,q)\mathscr{R}_a(t,s)$ dans le cas (b) et $(p,q)\mathscr{R}_a(s,q)$ dans le cas (c) indépendamment de la valeur de t.

En suivant la description faite dans la Fig. 3, il est alors possible de définir une relation d'adjacence[1] $\mathscr{R}_a \subset \delta(X) \times \delta(X)$ telle que chaque *lignel* de bord possède exactement 2 voisins par $\mathscr{R}_a$: un pour chacune de ses 2 extrémités. Les composantes connexes de $\delta(X)$ suivant $\mathscr{R}_a$ (c.-à.-d. les classes d'équivalences de la fermeture transitive de $\mathscr{R}_a$) forment alors l'ensemble des *contours* inter-pixels de X. Nous avons choisi la construction de $\mathscr{R}_a$ décrite par la Fig. 3 afin d'obtenir des contours qui séparent une composante 8-connexe de X de l'une des composantes 4-connexes de $\overline{X}$, au sens où tout 4-chemin de pixels entre ces deux ensembles *traverse* le contour. Nous considérons en effet les traits comme un ensemble de composantes 8-connexes de pixels non nuls. Enfin, moyennant le choix d'un *lignel* et d'une de ses 2 extrémités, il est possible de munir chaque cycle correspondant à un contour d'une paramétrisation complète $C = (a_0, \ldots, a_{n-1})$, $a_i \in \delta(X)$ où n est le nombre d'arêtes du contour et $a_i \mathscr{R}_a a_{i+1 \bmod n}$ pour tout $i \in \{0, \ldots, n-1\}$ (voir [8, Section 6]).

Normales canoniques : Pour tout *lignel* $e = (p,q)$ du bord de X, on désigne par *normale canonique en* e le vecteur unitaire $\mathbf{n}(e) = (x_q - x_p, y_q - y_p)$. L'ensemble des vecteurs normaux canoniques au bord d'un objet est représenté dans la Fig. 5a.

Normale estimée au bord de X : Soit $C = (a_0, \ldots, a_{n-1})$ une paramétrisation d'un contour de X et $i \in \{0, \ldots, n-1\}$. On estime le vecteur normal au contour en l'arête a_i, que l'on note $\tilde{\mathbf{n}}(a_i)$, par le vecteur $\tilde{\mathbf{n}}(a_i) = \frac{\mathbf{m}(a_i)}{\|\mathbf{m}(a_i)\|}$ avec :

$$\mathbf{m}(a_i) = \sum_{-5 \le k \le 5} e^{\frac{k^2}{30}} \mathbf{n}(a_{i+k \bmod n}) \tag{2}$$

Dans le cas où le contour C comporte moins de 11 *lignels*, la somme de l'équation (2) est restreinte de façon à prendre en compte une seule fois chaque arête. Cette estimation, basée sur un simple lissage des normales canoniques, est suffisament précise pour caractériser par la suite les points de forte courbure de traits dont l'épaisseur est de l'ordre de quelques pixels.

[1] Relation binaire symétrique et anti-réflexive.

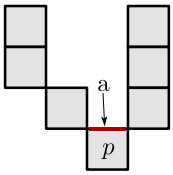


FIG. 4: Au pixel p, le bord de cette courbe de pixels présente une courbure à la fois négative (en a) et positive (aux autres *lignels* de p). On associe une courbure positive au pixel p en ne tenant pas compte de la courbure négative au *lignel* a (Éq. (5)).

Courbure estimée au bord de X : La courbure signée estimée au lieu de l'arête a_i, notée $\tilde{\kappa}(a_i)$, est calculée par :

$$\tilde{\kappa}(a_i) = \text{sign}(\det(\tilde{\mathbf{n}}(a_{i-1}), \tilde{\mathbf{n}}(a_{i+1}))) \frac{\|\tilde{\mathbf{n}}(a_{i+1}) - \tilde{\mathbf{n}}(a_{i-1})\|}{2} \tag{3}$$

Propriété 1 *La courbure ainsi définie vérifie* $|\tilde{\kappa}(a)| \le 1$.

Dans la suite, on appelle *pixel de bord* un pixel de X possédant une arête de bord (autrement dit, 4-adjacent à un pixel de $\overline{X}$).

Normale estimée aux pixels de bord : Soit p un pixel de bord, et $L(p)$ l'ensemble des *lignels* de bord de la forme $(p,\cdot)$. On définit $\tilde{\mathbf{n}}(p)$, le vecteur normal associé au pixel p, par $\tilde{\mathbf{n}}(p) = \frac{\tilde{\mathbf{m}}(p)}{\|\tilde{\mathbf{m}}(p)\|}$, avec :

$$\tilde{\mathbf{m}}(p) = \sum_{a \in L(p)} \tilde{\kappa}(a)^2 \cdot \tilde{\mathbf{n}}(a) \tag{4}$$

Courbure estimée aux pixels de bord : Nous associons à chaque pixel de bord p la courbure $\tilde{\kappa}(p)$ définie par :

$$\tilde{\kappa}(p) = \max_{a \in L(p)} \{\max(0, \tilde{\kappa}(a))\} \tag{5}$$

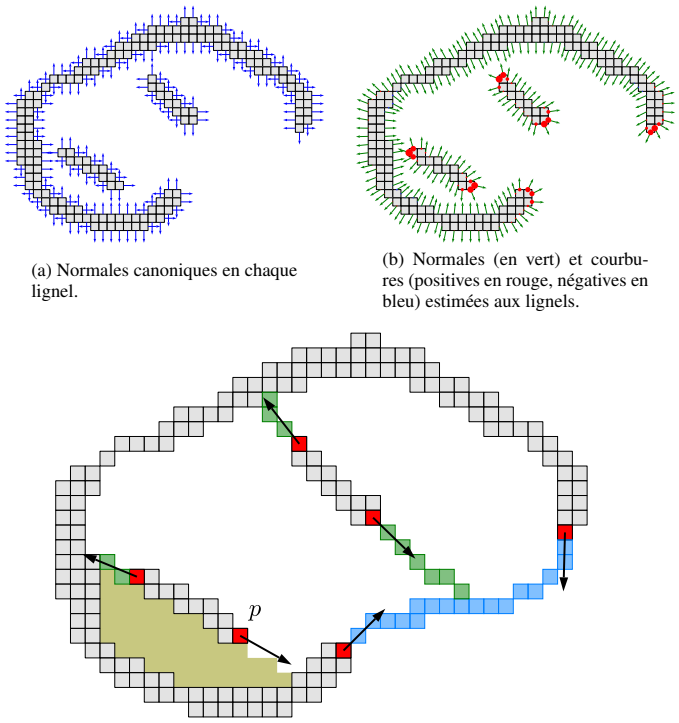
La courbure utilisée par la suite est ainsi définie comme nulle aux *lignels* de bord a pour lesquels $\tilde{\kappa}(a) < 0$. En effet, les points d'intérêt détectés par la suite étant caractérisés par une courbure positive extrême, les courbures négatives ne sont pas pertinentes. Par exemple, dans le cas de la Fig. 4, la courbure négative au *lignel* a situé à l'intérieur de la courbe de pixels ne doit pas intervenir dans le calcul de la courbure au pixel p. Seuls les trois autres *lignels*, ayant une courbure positive, sont pris en compte.

2.2.2 Ensemble $\mathcal{I}$ de points d'intérêt

Nous avons défini dans la section précédente le vecteur normal ainsi que la courbure associée aux pixels du bord de X. Nous pouvons maintenant définir un premier ensemble de points $\mathcal{I}'$ composé des pixels présentant une courbure positive extrême. Pour un seuil θ_κ vérifiant $0 < \theta_\kappa < 1$, on définit

$$\mathcal{I}' = \{p \in X, \tilde{\kappa}(p) \ge \theta_\kappa\}. \tag{6}$$

Finalement, l'ensemble $\mathcal{I}'$ n'étant pas nécessairement composé de pixels isolés, seul un pixel est conservé au sein de chacune de ses composantes 8-connexes : le pixel de courbure maximale sur chaque composante. L'ensemble de pixels ainsi obtenu est noté $\mathcal{I}$ (pixels en rouge de la Fig. 5c).

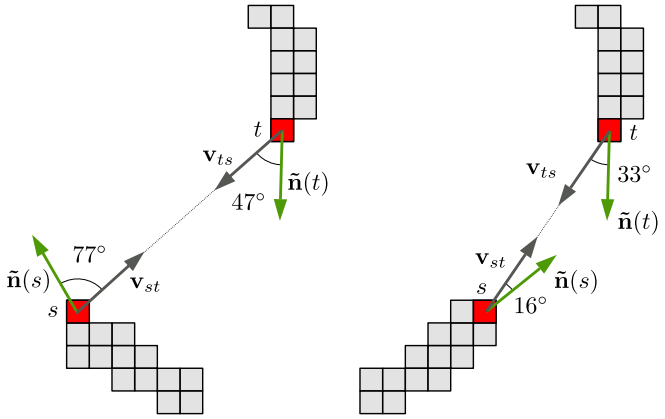


(a) Normales canoniques en chaque lignel.

(b) Normales (en vert) et courbures (positives en rouge, négatives en bleu) estimées aux lignels.

(c) Fermeture par des *splines* (en bleu) et des segments (en vert). Les points d'intérêt détectés sont représentés en rouge, avec leur vecteur normal estimé.

FIG. 5: Illustration de différentes étapes de l'algorithme de fermeture de traits.



(a) Configuration pour laquelle $\tilde{\mathbf{n}}(s)\mathbf{v}_{st} + \tilde{\mathbf{n}}(t)\mathbf{v}_{ts} = 0{,}9$.

(b) Configuration pour laquelle $\tilde{\mathbf{n}}(s)\mathbf{v}_{st} + \tilde{\mathbf{n}}(t)\mathbf{v}_{ts} = 1{,}8$.

FIG. 6: Illustration du second terme du facteur de qualité $\omega(s,t)$. La configuration (b) est jugée plus favorable à la fermeture par une courbe *spline* que la configuration (a).

2.3 Fermeture des traits

Munis de l'ensemble de points d'intérêts $\mathcal{I}$, nous proposons de fermer les traits de dessin en combinant deux méthodes :

- Rejoindre certaines paires de points d'intérêt par des courbes *splines* discrétisées (section 2.3.1). Ces dernières seront paramétrées par les normales estimées aux points considérés (voir la courbe bleue de la Fig. 5c).
- Prolonger certains traits en traçant des segments de droites issus d'un point d'intérêt, dans la direction de la normale estimée en ce point (section 2.3.2). La Fig. 5c comporte trois traits de ce type, en vert.

Dans la suite, nous appelons *trait de fermeture* une séquence de pixels formant une courbe *spline* ou un segment de droite et qui, comme décrit précédemment, peut être dessinée sur l'image sous certaines conditions.

2.3.1 Connexion par des *splines*

Dans un premier temps, nous décrivons les différentes étapes ou notions utilisées pour la connexion par *splines*, en commençant par le critère utilisé pour décider de relier deux points.

Définition d'un critère de connexion : Nous définissons un facteur de qualité associé à tout couple de points d'intérêt qui seront éventuellement réunis par une courbe *spline*. Cette valeur sera utilisée comme critère de rejet quand elle sera égale à zéro, mais aussi pour ordonner les configurations restantes. La méthode de fermeture proposée permet en effet d'autoriser, mais aussi d'interdire, les intersections entre les traits ajoutés. Dans le cas où les intersections sont interdites, il est donc important de dessiner en premier les traits jugés plus prioritaires. Ainsi, si s et t sont deux pixels de $\mathcal{I}$, la connexion des points d'intérêt par une courbe *spline* est paramétrée par plusieurs grandeurs, parmi lesquelles on trouve :

- $d_{\max}$, la distance limite à partir de laquelle deux points d'intérêts ne devront pas être reliés ;
- α, l'angle maximum entre les normales $\tilde{\mathbf{n}}(s)$ et $-\tilde{\mathbf{n}}(t)$.

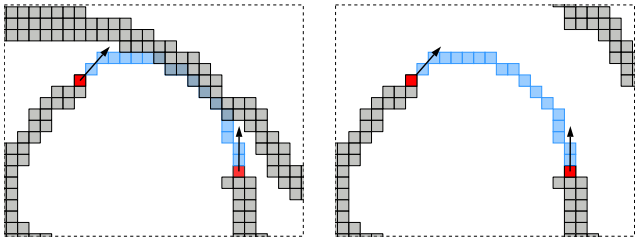
On définit alors le facteur de qualité associé au couple de pixels s et t, noté $\omega(s,t)$, par :

$$\begin{aligned}\omega(s,t) \quad = \quad & \max(0, 1 - \tfrac{\|s-t\|}{d_{\max}}) \\ & \cdot \max(0, \tilde{\mathbf{n}}(s)\mathbf{v}_{st} + \tilde{\mathbf{n}}(t)\mathbf{v}_{ts}) \\ & \cdot \max(0, \tilde{\mathbf{n}}(s) \cdot (-\tilde{\mathbf{n}}(t)) - \cos(\alpha))\end{aligned} \tag{7}$$

où $\mathbf{v}_{st} = \frac{t-s}{\|t-s\|}$ et $\mathbf{v}_{ts} = -\mathbf{v}_{st} = \frac{s-t}{\|s-t\|}$.

Remarque 1 *On a* $\omega(s,t) = \omega(t,s)$ *et* $0 \leq \omega(s,t) \leq 2$.

Comme mentionné précédemment, toute paire de points $\{s,t\}$ ne sera pas considérée dès lors que $\omega(s,t) = 0$. Le premier terme favorise les couples de points proches et exclut donc les points tels que $\|s-t\| \geq d_{\max}$. Le second terme favorise les couples dont la direction de la normale estimée en chacun des deux points est proche de la direction vers le point opposé (voir Fig. 6). Enfin, le troisième terme permet de favoriser les couples dont les vecteurs normaux estimés sont de sens opposés mais de directions proches, tout en excluant un écart de direction supérieur ou égal à α.
Finalement, l'ensemble des paires de pixels $\{s,t\}$ de $\mathcal{I}$ vérifiant $\omega(s,t) > 0$ est placé dans une liste $\mathcal{Q}_\omega(\mathcal{I})$ triée selon une priorité décroissante.



(a) Trait de fermeture invalide. (b) Trait de fermeture valide.

FIG. 7: Configurations de fermeture par courbe *spline* sur une partie d'image I_f (en gris). En (a) la courbe *spline* $\mathcal{S}$ (en bleu) issue des deux points d'intérêt (en rouge) est telle que $\tau(\mathcal{S}, I_f) = 4$, elle n'est donc pas ajoutée. En (b) on a $\tau(\mathcal{S}, I_f) = 2$, la courbe peut donc être dessinée.

Tracé de *splines* discrétisées : Une telle courbe est définie par deux points extrémités $\{s, t\}$ ainsi que deux vecteurs tangents en chacun de ces points. La distance entre les deux points s'avère être en pratique un facteur convenable, à appliquer aux vecteurs normaux estimés $\tilde{\mathbf{n}}(s)$ et $\tilde{\mathbf{n}}(t)$, pour obtenir les deux vecteurs tangents utilisés. Il est toutefois possible dans l'implémentation proposée de moduler l'aspect des *splines* produites en appliquant à cette distance un facteur supplémentaire ρ, avec $1 \leq \rho \leq 2$. Dans l'algorithme 2, la fonction `spline(`s,t,$\mathbf{v}_s$,$\mathbf{v}_t$`)` retourne la suite des points obtenus par discrétisation de la courbe *spline* définie par les deux pixels s et t, dont les dérivées respectives en ces points sont $\mathbf{v}_s$ et $\mathbf{v}_t$.

Gestion de la sur-segmentation du fond : La méthode de fermeture des traits proposée a pour effet de segmenter le fond de l'image binaire en régions 4-connexes ; régions qui sont par la suite colorisées automatiquement (section 3). Généralement, une sur-segmentation du fond est préjudiciable car elle rendrait pénible le travail du coloriste qui devrait affecter manuellement les couleurs définitives à de nombreuses régions de petite taille. Nous proposons donc d'éviter les intersections entre les traits de fermeture et l'image de traits binarisée (Fig. 7), mais aussi d'empêcher simplement la création de regions de petite taille.

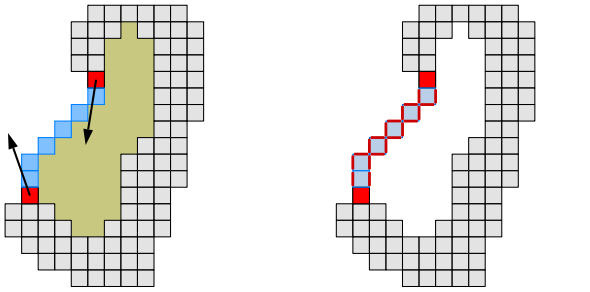
Gestion des intersections : Pour déterminer de manière fiable si un trait de fermeture intersecte une image binaire ailleurs qu'en ses extrémités, nous avons recours à la notion de nombre de transitions entre un chemin de pixels et une image. Ceci est justifié par l'existence de configurations pour lesquelles plusieurs pixels contigus, situés à l'extrémité d'une *spline*, font partie de l'image de traits.

Définition 1 *Soit J une image binaire définie sur Ω et $\mathcal{C} = (p_0, \ldots, p_{n-1})$ la paramétrisation d'un arc 8-connexe de pixels reliant p_0 à p_{n-1}, $n \in \mathbb{Z}$. On définit le* nombre de transitions *associé à $\mathcal{C}$ et J, que l'on note $\tau(\mathcal{C}, J)$, par :*

$$\tau(\mathcal{C}, J) = \left|\left\{i \in \{0, \ldots, n-2\} \text{ t.q. } J_{(p_i)} = 1 \veebar J_{(p_{i+1})} = 1\right\}\right|$$

où $\veebar$ désigne le « ou exclusif ».

Intuitivement, si $p_0, p_{n-1} \in \Omega$ vérifient $J_{(p_0)} = J_{(p_{n-1})} = 1$, on a $\tau(\mathcal{C}, J) = 2$ si et seulement si l'arc $\mathcal{C}$ quitte une seule fois les pixels non nuls de J pour y revenir ensuite une seule fois. Ainsi, un trait de fermeture $\mathcal{C}$ entre deux pixels ne sera pas ajouté à l'image binaire I_f (en cours de fermeture) si $\tau(\mathcal{C}, I_f) \neq 2$ dans le cas où l'on souhaite éviter les auto-intersections, et si $\tau(\mathcal{C}, I_b) \neq 2$ dans le cas contraire. Nous avons en effet choisi d'introduire un paramètre supplémentaire laissant la possibilité d'autoriser ou d'empêcher les intersections entre les traits de fermeture eux-mêmes (paramètre *auto-intersections*). Quand elles sont autorisées, le test d'intersection décrit précédemment est effectué sur l'image binaire d'entrée I_b, sinon il est fait sur l'image en cours de fermeture contenant les traits déjà ajoutés.



(a) Courbe *spline* créant une région de petite taille, en marron (48 pixels). (b) *Lignels* à partir desquel un suivi de contour est effectué.

FIG. 8: Détection de regions de trop petites tailles.

Creation de régions de taille convenable : Afin d'éviter la sur-segmentation, la méthode de fermeture proposée repose aussi sur un critère supplémentaire empêchant l'ajout d'un trait de fermeture lorsque celui-ci ferait apparaître, un fois dessiné dans l'image, au moins une composante 4-connexe de fond dont le nombre de pixels serait inférieur à un seuil a_{min} donné (cas de la region de couleur marron de la Fig. 8a si $a_{\text{min}} = 50$). Cependant, une région de moins de 5 pixels n'est pas considérée comme significative de ce point de vue en raison de l'apparition possible de régions de quelques pixels. C'est le cas par exemple au voisinage d'un point d'intérêt d'où sont issus plusieurs traits de fermeture. L'apparition de ces régions non significatives ne doit donc pas empêcher l'ajout d'un trait. Finalement, pour un trait candidat $\mathcal{T}$ composé de n pixels, nous avons recours à un algorithme de complexité $O(n)$ permettant de déterminer si toute région 4-connexe $R \subset \overline{J_b \cup \mathcal{T}}$ bordée en partie par $\mathcal{T}$ dans une image binaire J_b vérifie $(|R| \geq a_{\text{min}}) \vee (|R| < 5)$. Nous décrivons ici le fonctionnement de cet algorithme utilisé comme une fonction `RégionsValides(`J_b,$\mathcal{T}$,a_{min}`)`, utilisée dans l'algorithme 2. Cet algorithme repose sur un dessin préalable du trait avec une valeur spéciale (en affectant par exemple le second bit le moins significatif de la valeur de l'image) permettant par la suite de le retirer si besoin. Un suivi de contour est lancé, séquentiellement, depuis chaque lignel de bord du trait (lignels marqués en rouge sur la Fig. 8b), à moins qu'un autre parcours ait déjà visité le lignel en question. On utilise pour cela une procédure de suivi de contour d'un objet 8-connexe, autrement dit le parcours du bord d'une composante 4-connexe de son complé-

Algorithme 1 : `AireInterne`

Entrée : $\mathcal{C} = \{(p_i, q_i), 0 \leq i < N\}$, ensemble des *lignels* de bord de $Y \subset \Omega$
Sortie : $|Y|$
$a \leftarrow 0$
pour tous les $(p_i, q_i) = ((x_p, y_p), (x_q, y_q)) \in \mathcal{C}$ **faire**
  si $x_q = x_p + 1$ **alors** $a \leftarrow a + x_p + 1$
  sinon si $x_q = x_p - 1$ **alors** $a \leftarrow a - x_p$
fin
retourner a

mentaire. Cette procédure classique et validée [8, Théorème 6], découle immédiatement de la relation $\mathscr{R}_a$ décrite dans la Fig. 3. Chacun de ces parcours est arrêté dès lors que le nombre de lignels visités atteint $2(a_{\min} + 1)$ ou, bien sûr, si le lignel de départ est rencontré avant cela. Dans le premier cas, il est en effet possible d'affirmer grâce au corollaire 1 de la propriété 2 (Annexe A) que la région bordée par ce *lignel* possède au moins $a_{\min}$ pixels. Dans le second cas, on peut calculer la taille de la région bordée par le contour complet à l'aide de l'algorithme 1, et donc vérifier si cette taille est significative (c.-à-d. comporte plus de 5 pixels). S'il est à noter qu'un parcours des composantes connexes de pixels de fond bordées par le trait aurait présenté la même complexité, l'implémentation choisie a le mérite de ne nécessiter aucune structure de donnée (pile ou file) autre que l'image.
Finalement, la méthode de fermeture par des courbes *splines*, mettant en œuvre les étapes précédemment décrites, est résumée dans l'algorithme 2.

2.3.2 Connexion par des segments

Après avoir tracé les *splines* jugées valides au cours de l'étape précédente, des segments de droites issus de chacun des points d'intérêts sont éventuellement ajoutés. Ces segments ont vocation à prolonger un trait existant dans la direction donnée par la normale estimée, si un trait du dessin ou une *spline* se trouvent à proximité dans cette direction. Comme pour les *splines*, le tracé de ces traits est conditioné à la vérification de plusieurs contraintes :

- $s_{\max}$, le longueur maximum du segment ;
- $c_{\max}$, nombre maximum de *splines* ou segments issus d'un même point d'intérêt ;
- $a_{\min}$, nombre minimum de pixels des régions créées par l'ajout d'un segment.

Finalement, cette étape décrite dans l'agorithme 3, utilise une fonction `segment` qui retourne un ensemble de coordonnées de pixels correspondant au tracé, depuis une origine et dans une direction donnée, d'une demi-droite jusqu'à ce que celle-ci rencontre une trait existant. Un ensemble vide est retourné si la distance $s_{\max}$ est parcourue sans atteindre de trait.
Les segments ajoutés par l'algorithme 3 sont représentés en vert dans la Fig. 5c. On remarque notamment sur cette figure l'effet de la condition portant sur la taille minimale des régions

Algorithme 2 : `FermerSplines`

Entrées : I_b, image des traits binarisés (section 2.1) ;
$\mathcal{Q}_\omega(\mathcal{I})$, file de priorité des paires de pixels $\{s, t\}$ de $\mathcal{I}$ vérifiant $\omega(s, t) > 0$, ordonnée selon $\omega(s, t)$;
$c_{\max}$, nombre maximum de courbes ou segments issus d'un même point d'intérêt ;
$a_{\min}$, nombre minimum de pixels des régions créées ;
$\rho \in [1, 2]$, facteur d'aspect des *splines* ;
auto-intersections, booléen.
Sorties : I_{splines}, image avec ajout de *splines*.
Données : $L(p)$, nombre de courbes issues de tout point $p \in \mathcal{I}$ (initialisé à zéro) ;
Recopier I_b dans I_{splines}
pour tous les $\{s, t\}$ *de* $\mathcal{Q}_\omega(\mathcal{I})$ **faire**
  $\delta \leftarrow \rho\|s - t\|$
  $\mathcal{S} \leftarrow$ `spline(`s,t,$\delta\tilde{\mathbf{n}}(s)$,$\delta\tilde{\mathbf{n}}(t)$`)`
  si $L(s) < c_{max}$ *et* $L(t) < c_{max}$ **alors**
    si *auto-intersections* **alors**
      $t \leftarrow \tau(\mathcal{S}, I_b)$
    sinon
      $t \leftarrow \tau(\mathcal{S}, I_f)$
    fin
    si $t = 2$ *et* `RégionsValides(`$I_{splines}$,$\mathcal{S}$,a_{min}`)` **alors**
      pour tous les $(x, y) \in \mathcal{S}$ **faire**
        $I_{\text{splines}}(x, y) \leftarrow 1$
      fin
      $L(s) \leftarrow L(s) + 1$
      $L(t) \leftarrow L(t) + 1$
    fin
  fin
fin
retourner $I_{splines}$

créées. En effet, la région colorée en marron comporte 50 pixels ; ce qui, pour un paramètre $a_{\min} > 50$ empêche le dessin d'un segment issu du pixel p.
En résumé, la méthode de fermeture de traits consiste, après caractérisation des points d'intérêt de l'image I_b, en l'application de l'algorithme 2 pour obtenir une image I_{splines} ; image sur laquelle on applique l'agorithme 3 pour enfin obtenir l'image de traits fermés I_f.

3 Colorisation des aplats

L'obtention de l'image binarisée I_f des traits fermés est à la base du processus de colorisation du dessin I proprement dit. Nous proposons ici deux méthodes distinctes permettant d'obtenir un calque de couleur $C : \Omega \rightarrow [0, 255]^3$ associé au dessin I, qui correspondent à deux schémas de travail valables pour le coloriste.

1. Colorisation aléatoire des zones connexes : L'idée est de générer un calque de colorisation C composé de régions de couleurs aléatoires, et constantes par morceaux, telles que

Algorithme 3 : `FermerSegments`

Entrées : I_{splines}, image de traits fermés par des *splines* (section 2.3.1) ;
$\mathcal{I}$, liste des points d'intérêt ;
$c_{\max}$, nombre maximum de courbes ou segments issus d'un même point d'intérêt ;
$s_{\max}$, longueur de segment maximum ;
$L(p)$, nombre de courbes issues de tout point $p \in \mathcal{I}$ (après application de l'algorithme 2)
Sorties : I_{f}, image fermée.
Recopier I_{splines} dans I_f
pour tous les p *de* $\mathcal{I}$ **faire**
  si $L(p) < c_{max}$ **alors**
    $\mathcal{S} \leftarrow$ `segment` $(p, \tilde{\mathbf{n}}(p), s_{\max},)$
    si $\mathcal{S} \neq \emptyset$ *et* `RégionsValides` $(I_f, \mathcal{S}, a_{min})$ **alors**
      pour tous les $q \in \mathcal{S}$ **faire**
        $I_f(q) \leftarrow 1$
      fin
    fin
  fin
fin
retourner I_f

ces régions forment un sur-partitionnement du dessin d'origine (Fig. 10b). Le travail du préparateur d'aplats se résumera donc par la suite à assigner une couleur crédible à chaque région de C, la même couleur pouvant être *in-fine* attribuée à plusieurs régions voisines de C (Fig. 10c). Pratiquement, cela se réalise avec l'outil *remplissage* (*bucket fill*) des logiciels de dessin numériques, appliqué directement sur le calque C.
La génération de C est ici réalisée par une labelisation en composante connexes de l'image binaire I_f avec l'algorithme rapide [9], (de complexité linéaire), pour tous les points $I_{f(x,y)} = 0$. Dans un second temps, ces labels sont propagés pour les points $I_{f(x,y)} = 1$, par un algorithme de propagation par queue de priorité (*watershed*) [10], en utilisant une priorité P définie comme

$$\forall (x,y) \in \Omega, \quad P_{(x,y)} = - \min_{(p,q)\in\Omega} \left[\|(x,y)-(p,q)\| \mid I_{f(p,q)} = 1 \right]$$

aboutissant à une reconstruction des labels sur les contours résiduels, en *pelure d'oignon*. Le calcul de P se réalise grâce à l'algorithme rapide [5] de calcul de la fonction distance. On attribue ensuite une couleur aléatoire à chaque label différent obtenu. La fermeture préalable des traits de dessin permet ainsi d'obtenir de nombreuses composantes connexes, et un sur-partionnement judicieux du dessin d'origine I (voir par exemple la zone frontière ouverte front-cheveux en Fig. 10e).

2. Colorisation guidée par touches de couleurs : Cette deuxième technique de colorisation reprend l'idée de l'extrapolation de touches de couleurs, placées par l'utilisateur (Fig. 10f), pour former une partition du dessin entier (Fig. 10g). Cette idée est commune aux travaux de l'état de l'art [1, 2, 3, 4]. Là encore, la propagation des pixels labelisés par l'utilisateur se réalise par une propagation *watershed* [10], avec une priorité P définie à partir de l'image des traits de dessin fermés I_f :

$$\forall (x,y) \in \Omega, \quad P_{(x,y)} = \min((I_f * G_{\sigma_1})_{(x,y)}, (I_f * G_{\sigma_2})_{(x,y)})$$

où $I_f * G_\sigma$ dénote la convolution de I_f par un noyau gaussien 2d isotrope d'écart type σ. Nous considérons deux échelles d'analyse locales/globales

$$\sigma_1 = 1 \qquad \text{et} \qquad \sigma_2 = \max(l,h)/100$$

pour estimer une fonction potentielle P continue de l'image I_f. Là encore, la prise en compte des contours fermés pour la propagation des taches de couleurs joue un rôle prépondérant dans l'efficacité de l'algorithme de colorisation (La Fig. 10g,i illustre les résultats obtenus pour deux potentiels P calculés à partir de I_f et de I, c.-à-d. avec et sans fermeture des traits).

4 Analyse et conclusion

Les Fig. 9,10 résument l'utilisation des deux méthodes proposées de colorisation semi-guidée par aplats, et montre très clairement que la fermeture des traits de dessin est une étape essentielle pour l'obtention de résultats pertinents. La Fig. 11 quant-à-elle illustre l'influence des différents paramètres de l'algorithme, pour la colorisation de l'image du triangle de *Kanizsa*. En faisant varier les longueurs maximales autorisées des *splines* et des segments, et des aires maximales des régions, nous sommes capable de générer des résultats de colorisation variés, et surtout qui sont tous cohérents. Cela signifie que l'on est en pratique capable de s'adapter à de nombreux types de dessins, et à des résolutions d'images différentes. Par ailleurs, en évitant un processus de minimisation coûteux en temps de calcul, et en utilisant des techniques de labelisation/propagation de complexité quasi-linéaires, notre méthode s'avère particulèrement rapide et simple à implémenter en comparaison avec les algorithmes de l'état de l'art. La comparaison du temps de calcul de notre algorithme avec une implémentation similaire (uniquement *CPU*, et en *C++*) de la technique *LazyBrush* proposée dans [4], montre un gain de 92% (*4.8s* contre *57s*) sur une image de résolution 2630×1200).
Notons également que notre technique de colorisation fonctionne de manière satisfaisante avec les images de dessin I contenant des traits *anti-aliasées*, bien que l'analyse géométrique proposée s'effectue sur une image binarisée I_b. La raison est double : d'une part, la binarisation de I ne déteriore pas fondamentalement la structure géométrique des traits de dessin, et les traits éventuellement perdus (morceaux de traits clairs qui pourraient passer sous le seuil s'il est mal choisi par exemple) sont de toute façon reconstruits par l'algorithme de fermeture. D'autre part, les méthodes de propagation utilisés dans le processus de génération du calque de colorisation C assurent que des couleurs soient effectivement reconstruites « sous les traits », c.-à-d. à des localisations de pixels de traits de dessin. La fusion de I et de C (par multiplication) permet donc bien de coloriser légèrement les pixels appartenant aux traits qui ne

seraient pas complètement noir, mais faisant partie de l'anti-aliasage.
Pour finir, il est à noter que dans un souci de reproductibilité scientifique, notre algorithme a été intégré dans un cadriciel libre de traitement d'image [11], ce qui permet son utilisation par le plus grand nombre à des fins applicatives ou de comparaison.

References

[1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 689–694, ACM, 2004.

[2] J. Qiu, H. S. Seah, F. Tian, Z. Wu, and Q. Chen, "Feature- and region-based auto painting for 2d animation," *The Visual Computer*, vol. 21, no. 11, pp. 928–944, 2005.

[3] Y. Qu, T.-T. Wong, and P.-A. Heng, "Manga colorization," in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 1214–1220, ACM, 2006.

[4] D. Sỳkora, J. Dingliana, and S. Collins, "Lazybrush: Flexible painting tool for hand-drawn cartoons," in *Computer Graphics Forum*, vol. 28, pp. 599–608, Wiley Online Library, 2009.

[5] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *Proceedings of the 5th International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing, ISMM 2000, Palo Alto, CA, USA, June 26-28, 2000*, pp. 331–340, 2000.

[6] S. Fourey and R. Malgouyres, "Normals estimation for digital surfaces based on convolutions," *Computers & Graphics*, vol. 33, no. 1, pp. 2–10, 2009.

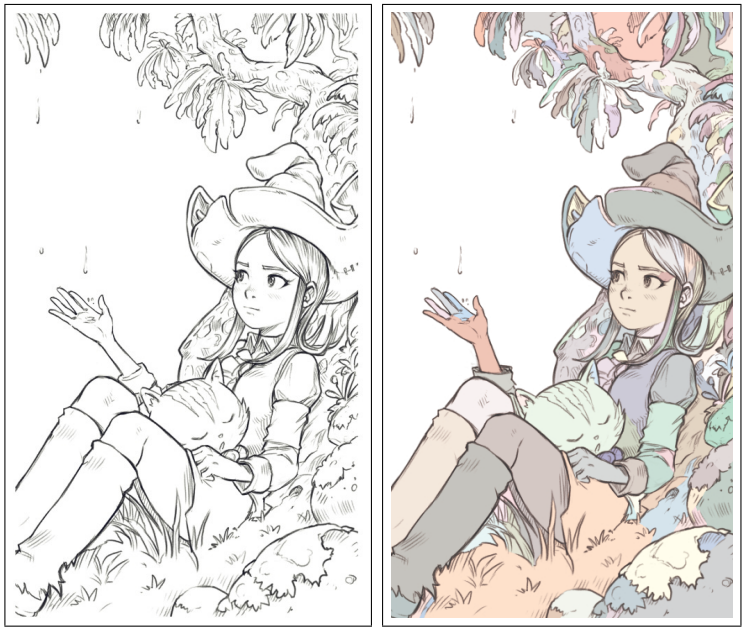
[7] A. Rosenfeld, "Adjacency in digital pictures," *Information and Control*, vol. 26, no. 1, pp. 24–33, 1974.

[8] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.

[9] W. H. Hesselink, A. Meijster, and C. Bron, "Concurrent determination of connected components," *Science of Computer Programming*, vol. 41, no. 2, pp. 173 – 194, 2001.

[10] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," *OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED-*, vol. 34, pp. 433–433, 1992.

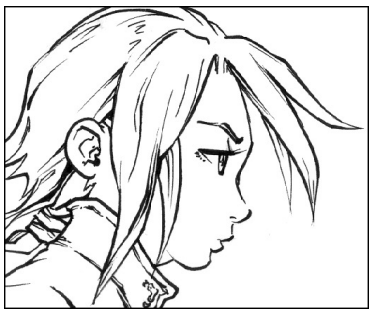
[11] D. Tschumperlé, "G´MIC: GREYC´s Magic for Image Computing: A full-featured open-source framework for image processing." `http://gmic.eu/`, 2008–2017.



(a) Résultat de notre algorithme de colorisation de région, en couleurs aléatoires.

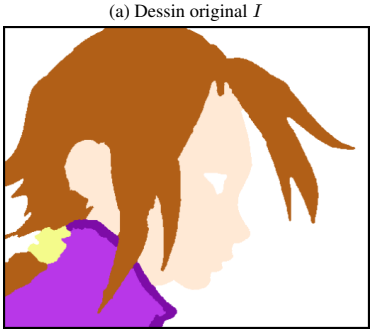
(b) Post-traitements manuels effectués par l'artiste pour arriver au rendu final

FIG. 9: Etapes de colorisation d'un dessin au trait : A partir de la colorisation automatique *(a)* produite par notre algorithme, l'artiste affecte les couleurs pour chaque région générée *(b,gauche)*, puis ajoute ombres, textures, effets de lumières et corrections colorimétriques, pour l'obtention du résultat final *(b,droite)*.

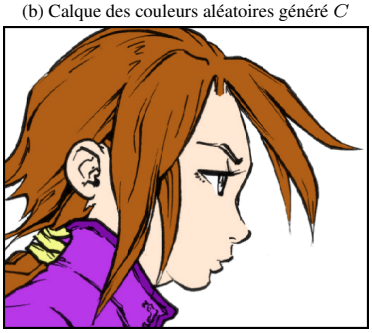


(a) Dessin original I

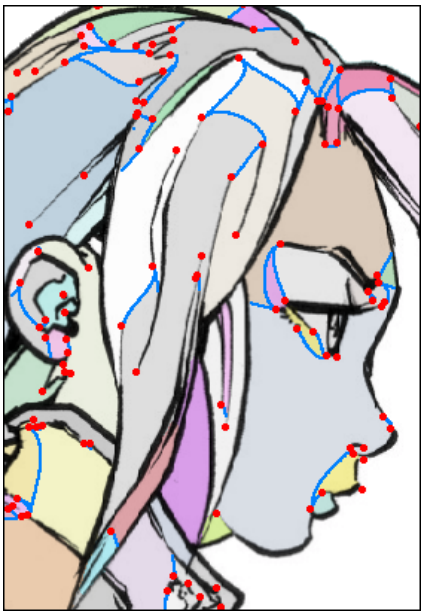
(b) Calque des couleurs aléatoires généré C



(c) Affectation des couleurs d'aplats par l'artiste



(d) Résultat : dessin colorisé $I * C$

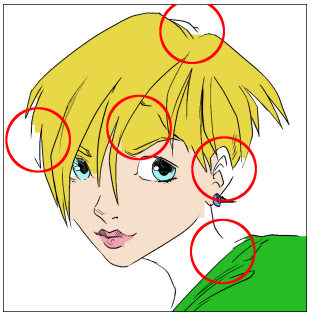


(e) Points d'intérêts détectés (en rouge), et courbes ajoutées (en bleu) par l'algorithme pour la fermeture des régions dans I_f

(f) Dessin original + marqueurs de couleurs ajoutés par l'utilisateur

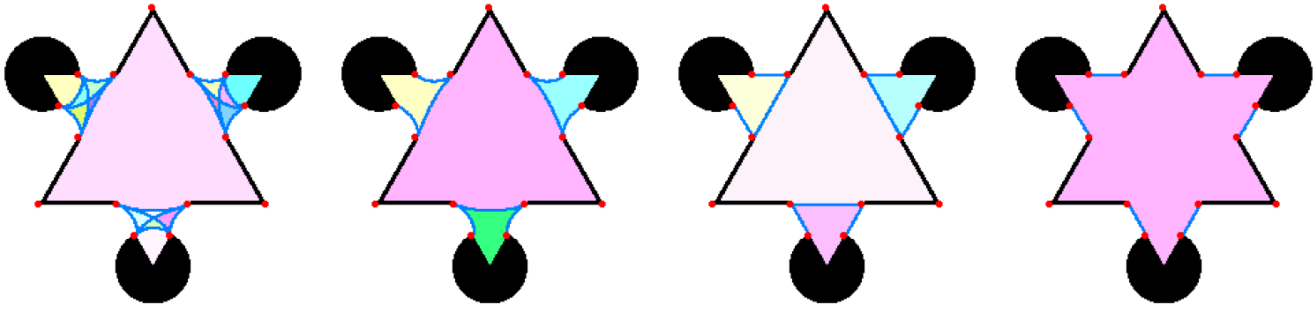
(g) Propagation des couleurs après fermeture des régions

(h) Dessin colorisé



(i) Résultat obtenu sans fermeture des régions (défauts cerclés en rouge)

FIG. 10: Illustration des deux méthodes de colorisation proposées. ***Méthode 1 (a,b,c,d,e) :*** *Colorisation par affectation manuelle de couleurs sur un calque de couleurs aléatoires sur-segmenté.* ***Méthode 2 (f,g,h,i) :*** *Colorisation par propagation de marqueurs colorés définis par l'utilisateur.*



(e) (f) (g) (h)

FIG. 11: Influence des paramètres de l'algorithme de fermeture de régions : *(a) Splines courbes et aire minimale faible, (b) Splines courbes et aire minimale moyenne, (c) Splines droites et aire minimale moyenne, (d) Splines droites et aire minimale élevée.*

Appendices

A Algorithme `RégionsValides` : justification de la borne utilisée

Dans l'implémentation de l'algorithme `RégionsValides`, le nombre d'étapes du suivi initié en chaque lignel est borné par la constante $2(a_{\min}+1)$, où $a_{\min}$ est le nombre minimal de pixels des régions créées, d'où une complexité globale linéaire par rapport au nombre de pixels du trait candidat. Cette borne découle du lien entre le nombre de lignels du *contour externe* d'une région 4-connexe et son nombre de pixels. Nous énonçons ici ce lien et nous en donnons la preuve.
Nous définissons tout d'abord la notion de contour externe.

Définition 2 *Si E est une partie finie et 4-connexe de $\mathbb{Z}^2$, on appelle* contour externe *de E, que l'on note $\hat{\delta}(E)$, l'ensemble des* lignels *de $\delta(E)$ qui sont partagés par un pixel de E et un pixel de la composante 8-connexe infinie de $\mathbb{Z}^2 \setminus E$.*

On remarque que pour toute partie finie F de $\mathbb{Z}^2$, on a $\hat{\delta}(F) \subset \delta(F)$. Par conséquent,

$$|\hat{\delta}(F)| \leq |\delta(F)| \tag{8}$$

On peut maintenant énoncer la propriété 2 ainsi que son corrolaire qui valide l'algorithme utilisé.

Propriété 2 *Soit E une partie 4-connexe de Ω, on a :*

$$|\delta(E)| \leq 2|E| + 2 \tag{9}$$

Le corollaire suivant est une conséquence immédiate de cette propriété et de (8).

Corollaire 1 *Soit E une partie 4-connexe de Ω, on a :*

$$\frac{|\hat{\delta}(E)| - 2}{2} \leq |E| \tag{10}$$

On démontre la propriété 2 par récurence sur le nombre de pixels de l'ensemble E. Dans cette preuve, on aura recours au lemme suivant pour garantir la connexité lors de l'application de l'hypothèse de récurence.

Lemme 1 *Tout ensemble 4-connexe E possède au moins un pixel p tel que $E \setminus \{p\}$ est lui aussi 4-connexe.*

Preuve: Considérons le graphe $\mathcal{G}$ dont les sommets sont les pixels de E et les arêtes sont données par la relation de 4-adjacence entre pixels. Soit $\mathcal{A}$ un arbre couvrant de $\mathcal{G}$. Tout pixel p associé à une feuille de $\mathcal{A}$ est tel que $E \setminus \{p\}$ est 4-connexe. □

Remarque 2 *Pour tout ensemble fini de pixels E, le nombre de* lignels *de $\delta(E)$ est pair. En effet, les* lignels *verticaux (resp. horizontaux) peuvent êtres appariés de manière évidente.*

Le lemme suivant localise la contribution d'un pixel à l'apparition de lignels de bord, lors qu'un pixel est ajouté à un ensemble. Il permettra dans la preuve de majorer le nombre de lignels ajoutés.

Lemme 2 *Soit E un ensemble 4-connexe et $p \in E$. Si $\{a, b, c, d\}$ est l'ensemble des pixels 4-adjacents à p, on a :*

$$\delta(E) \setminus \delta(E \setminus \{p\}) \subset \{(p,a),(p,b),(p,c),(p,d)\}$$

Preuve: Soit $(r,s) \in \delta(E) \setminus \delta(E \setminus \{p\})$.
Supposons que $r \neq p$. Comme $(r,s) \in \delta(E)$, on a $r \in E$. Par conséquent, $r \in E \setminus \{p\}$. De plus, (r,s) étant dans le bord $\delta(E)$, on a $s \in \overline{E}$ et, comme $\overline{E} \subset \overline{E \setminus \{p\}}$, on en déduit que $s \in \overline{E \setminus \{p\}}$.
Or, si $r \in E \setminus \{p\}$ et $s \in \overline{E \setminus \{p\}}$, il s'ensuit que $(r,s) \in \delta(E \setminus \{p\})$. Ceci entre en contradiction avec le fait que $(r,s) \in \delta(E) \setminus \delta(E \setminus \{p\})$. Aussi, on obtient que $r = p$. □

Preuve de la propriété 2: On montre (9) par récurence sur $n = |E|$. Si $|E| = 1$, alors $|\delta(E)| = 4$ et l'inégalité est satisfaite pour $n = 1$. On suppose donc l'inégalité satisfaite pour tout ensemble 4-connexe E tel que $|E| = n$.
Considérons maintenenant un ensemble E' de pixels 4-connexe tel que $|E'| = n+1$. D'après le Lemme 1, il existe nécessairement un pixel $p \in E'$ tel que $E' \setminus \{p\}$ est 4-connexe. L'ensemble E' étant 4-connexe, ce pixel p partage nécessairement une arête avec un pixel q de $E' \setminus \{p\}$. Comme $p, q \in E'$, on en déduit que $(p,q) \notin \delta(E')$, et d'après le Lemme 2, que

$$|\delta(E') \setminus \delta(E' \setminus \{p\})| \leq 3 \tag{11}$$

D'autre part, pour deux ensembles A et B on a :

$$|A| \leq |B| + |A \setminus B| \tag{12}$$

D'après (11) on a donc :

$$|\delta(E')| \leq |\delta(E' \setminus \{p\})| + 3 \tag{13}$$

Or, le cas de l'égalité est exclu par la remarque 2). Finalement,

$$|\delta(E')| \leq |\delta(E' \setminus \{p\})| + 2. \tag{14}$$

Par hypothèse de récurence, l'ensemble $E' \setminus \{p\}$ étant 4-connexe et tel que $|E' \setminus \{p\}| = n$, on a :

$$|\delta(E' \setminus \{p\})| \leq 2n + 2 \tag{15}$$

De (15) et (14) on obtient :

$$|\delta(E')| \leq 2 + (2n+2) = 2(n+1) + 2 = 2|E'| + 2$$

La propriété est vraie pour un ensemble E' de taille $n+1$, elle l'est donc pour tout $n > 0$. □

Soit Y une composante 4-connexe du complémentaire de $X' \subset \Omega$ et soit $(p,q) \in \delta(X')$ tel que $q \in Y$. Par la propriété 2, il est possible de déterminer par un suivi du contour externe de Y, en partant du *lignel* (p,q), que le nombre de pixels de Y est supérieur à $a_{\min}$ dès lors que le nombre l de *lignels* parcourus vérifie $\frac{l-2}{2} = a_{\min}$, autrement dit :

$$l = 2(a_{\min} + 1) \tag{16}$$

Ceci valide le critère d'arrêt du suivi utilisé dans l'algorithme `RégionsValides` décrit dans la section 2.3.1.