

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

Sébastien FOUREY¹, David TSCHUMPERLÉ¹, David REVOY²

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

²Artiste 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), en deux étapes : 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.

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.

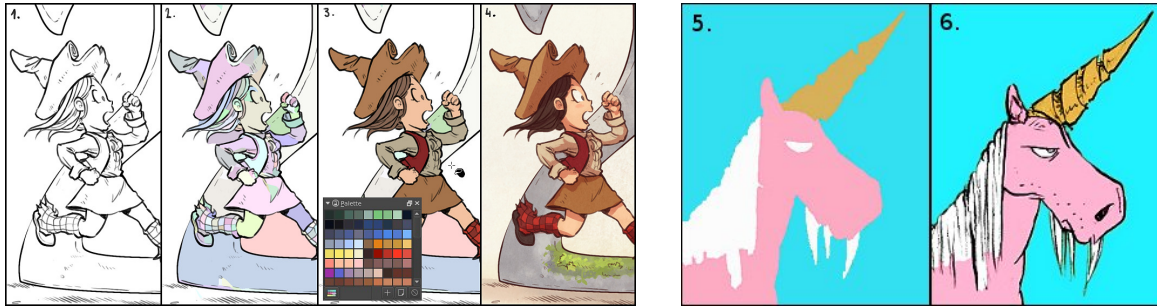


FIG. 1: Rôle et technique de la colorisation de dessins *en aplats* pour la bande dessinée.

1 Introduction

Dans le domaine de la bande dessinée, la colorisation d’un dessin au trait se réalise en deux étapes successives : Le dessin est d’abord pré-colorisé par le *préparateur d’aplats* (Fig. 1.3), en attribuant une couleur unique à chaque région distincte du dessin d’origine. Dans un second temps, le *coloriste* ajoute ombres, lumières et ambiance colorimétrique, pour obtenir le résultat de colorisation final (Fig. 1.4). La colorisation en aplats consiste donc à générer un calque qui ne contient que des zones de couleurs constantes par morceaux, formant une partition colorisée du plan (Fig. 1.5). Ce calque est fusionné avec les traits d’origine pour un rendu colorisé en aplats (Fig. 1.6). La colorisation en aplats est en pratique un processus fastidieux car les outils « classiques » des logiciels de peinture numérique ne rendent pas cette tâche facile : Les outils de remplissage par croissance de région gèrent mal les discontinuités dans les traits de dessin, surtout avec des traits *anti-aliasés*. Il est donc courant que l’artiste effectue ses aplats en peignant « manuellement » ses couleurs sur un calque séparé, avec tous les problèmes de précision aux abords des traits

que cela suppose. Il arrive même qu’il décide de contraindre son style de dessin, en utilisant par exemple des lignes crenelées en résolution supérieure plutôt qu’*anti-aliasées*, ou en se forçant à ne tracer que des traits sans « trous », pour faciliter le travail du préparateur d’aplats. Récemment, quelques algorithmes dédiés à la colorisation en aplats ont été proposés, d’abord en photographie [1], puis plus spécifiquement pour des dessins aux traits [2, 3, 4]. Toutes ces approches se basent sur des étapes similaires : l’utilisateur doit placer des marqueurs colorés sur son dessin, marqueurs qui sont ensuite extrapolés pour tous les pixels par la *minimisation d’une énergie globale* prenant en compte la géométrie du dessin. Malheureusement, la complexité algorithmique de ces minimisations est grande, et l’interaction utilisateur s’en trouve pénalisée, notamment pour des images à haute résolution (planches A4 de 5000×7000). Dans cet article, nous proposons un algorithme original de colorisation en aplats, donnant des résultats de qualité similaires à ces techniques d’optimisation, mais avec une complexité algorithmique moindre, permettant une meilleure interactivité 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 (trous), 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 colorés placés par l'utilisateur.

2 Fermeture de traits binarisés

On considère une image de dessin au trait $I : \Omega \rightarrow [0, 255]$ (en niveau de gris et de taille $w \times h$) que l'on souhaite coloriser, définie sur le domaine discret $\Omega = \{0, \dots, w-1\} \times \{0, \dots, h-1\}$. Le dessin est réalisé a 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, réalisée sur une version binarisée de I , dénotée $I_b : \Omega \rightarrow \{0, 1\}$.

Binarisation de l'image anti-aliasée : Pour $\theta \in [0, 255]$, on définit l'image binarisée I_b par simple seuillage de I : $I_b(x, y) = (I(x, y) \geq \theta)$, à valeur dans $\{0, 1\}$. Le problème se simplifie alors à celui de la fermeture de traits dans une image binaire, sans perte de généralité ni d'efficacité de la méthode proposée (voir section 4).

Caractérisation des points d'intérêt : La méthode repose sur la détection de point d'intérêts situés aux extrémités des traits de dessin, et sur l'estimation d'une direction convenable pour éventuellement prolonger ces traits (Fig. 2c). Nous proposons de caractériser ces points d'intérêt en estimant le champ des normales et courbures autour des zones non nulles de I_b par une méthode inspirée de [5], qui repose sur un moyennage géométrique local et qui est décrite ci-après.

Estimation des normales et courbures aux bords L'approche utilisée, dite inter-pixels, considère des courbes d'arêtes partagées par un pixel à valeur non nulle et un pixel 4-adjacent au premier et à valeur nulle. De telles arêtes sont appelées *lignels* de bord, et un algorithme de suivi classique et validé ([6, Théorème 6]) permet de former des courbes de *lignels* (*bi-curves* dans [7]) qui constituent les bords (resp. le bord) d'une région connexe (resp. simplement connexe) de pixels non nuls. Formellement, un lignel étant assimilable à un couple de pixels (p, q) tel que $I_b(p) = 1$ et $I_b(q) = 0$, $p, q \in \Omega$, il possède une normale canonique $\mathbf{n} = q - p$ (en bleu sur la figure 2a) définissant un champ de normales sur chaque courbe, à laquelle nous pouvons appliquer un moyennage gaussien pour obtenir une estimation des normales $\tilde{\mathbf{n}}(a)$ en chaque *lignel* a d'un contour de la forme (en vert sur la figure 2b). Ces normales estimées permettent à leur tour d'associer au *lignels* une courbure $\tilde{\kappa}(a)$ (illustrée en rouge et bleu sur la Fig. 2b), définie par

$$\tilde{\kappa}(a) = \text{sign}(\det(\tilde{\mathbf{n}}(a_p), \tilde{\mathbf{n}}(a_s))) \frac{\|\tilde{\mathbf{n}}(a_s) - \tilde{\mathbf{n}}(a_p)\|}{2} \quad (1)$$

où a_p et a_s sont les *lignels* précédant et suivant a sur le contour, respectivement.

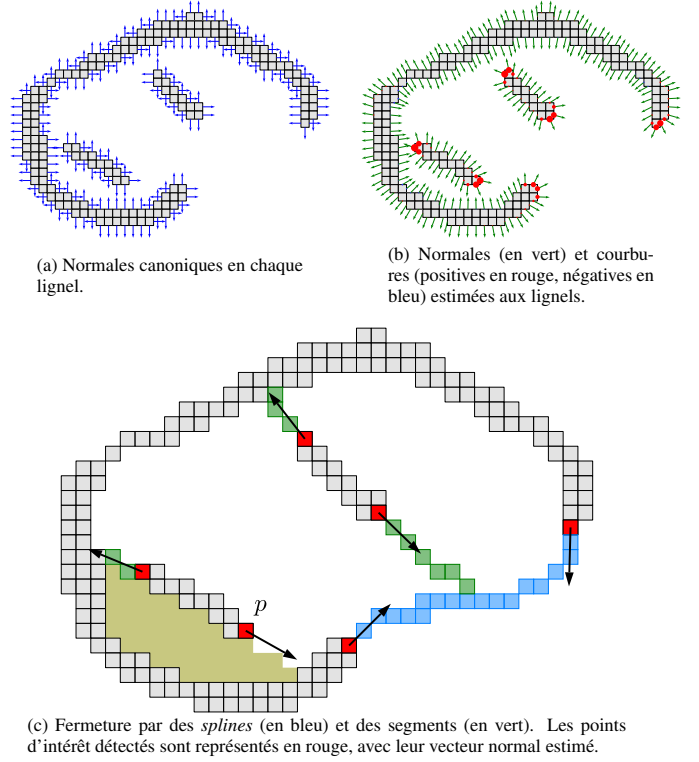


FIG. 2: Illustration de différentes étapes de l'algorithme de fermeture de traits.

Normales et courbures estimées associées aux pixels de bord La normale estimée associée à un pixel de bord p , aussi notée $\tilde{\mathbf{n}}(p)$, est alors définie par $\tilde{\mathbf{n}}(p) = \frac{\tilde{\mathbf{m}}(p)}{\|\tilde{\mathbf{m}}(p)\|}$, où $\tilde{\mathbf{m}}(p) = \sum_{a \in L(p)} \tilde{\kappa}(a)^2 \cdot \tilde{\mathbf{n}}(a)$, $L(p)$ désignant l'ensemble des *lignels* de bord du pixel p .

Finalement, on définit $\tilde{\kappa}(p)$, la courbure estimée associée à un pixel de bord p , par $\tilde{\kappa}(p) = \min_{a \in L(p)} \{\max(0, \tilde{\kappa}(a))\}$. Les courbures négatives ne sont ainsi pas prises en compte, puisque seules les extrémités des traits (associées à des courbures positives extrêmes, illustrées par des disques rouges sur la Fig. 2b) constituent des points d'intérêt.

La liste de points d'intérêt $\mathcal{I}$ est enfin définie comme l'ensemble des pixels p vérifiant $\tilde{\kappa}(p) \geq \theta_\kappa$, pour un seuil $\theta_\kappa \in]0, 1[$. Liste dont on ne conserve que le pixel de courbure maximum pour chaque composante connexe de points retenus (pixels en rouge sur la Fig. 2c).

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 par des courbes *splines* discrétisées, paramétrées par les normales estimées aux points considérés (courbe bleue de la Fig. 2c).
- 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. La figure 2c comporte trois traits de ce type, en vert.

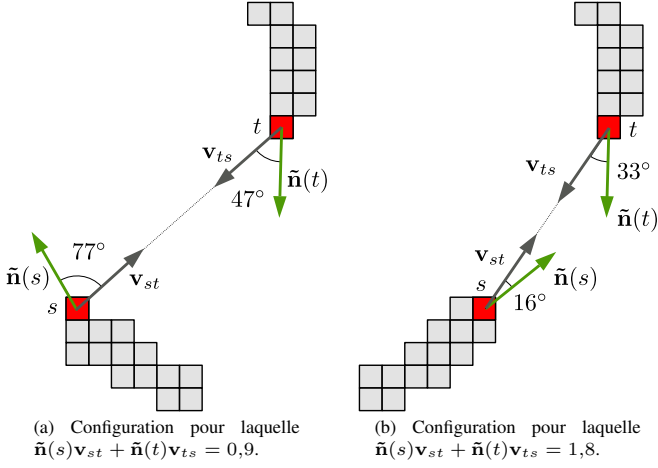


FIG. 3: Illustration du second terme du facteur de qualité $\omega(s, t)$. La configuration (b) est jugée plus favorable à la fermeture par une courbe *spline* que la configuration (a).

Dans la suite, *splines* et segments de droites sont appelés *traits de fermeture*.

Connexion par des splines : La fermeture par des courbes *splines* repose sur la définition d'un critère de connexion entre deux points d'intérêt, permettant d'affecter une priorité à l'ensemble de ces paires, d'autres étant simplement exclues (priorité nulle). Il est important d'ordonner les paires candidates car l'intersection entre *splines* pouvant être exclue, il convient de dessiner en premier les traits jugés plus prioritaires. Ainsi, si $s, t \in \mathcal{I}$, la connexion de s à t 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 ; et α , l'angle maximum entre les normales estimées en ces deux points. On définit alors le facteur de qualité associé au couple de pixels s et t , noté $\omega(s, t)$ et vérifiant $0 \leq \omega(s, t) \leq 2$, par :

$$\omega(s, t) = \omega(t, s) := \max(0, 1 - \frac{\|s - t\|}{d_{\max}}) \cdot \max(0, \tilde{\mathbf{n}}(s) \cdot \mathbf{v}_{st} + \tilde{\mathbf{n}}(t) \cdot \mathbf{v}_{ts}) \cdot \max(0, \tilde{\mathbf{n}}(s) \cdot (-\tilde{\mathbf{n}}(t)) - \cos(\alpha)) \quad (2)$$

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

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. 3). 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 .

Les paires de points sont alors parcourues dans l'ordre de priorité décroissante, et des *splines* discrétisées sont dessinées, ayant pour dérivées aux extrémités les normales estimées pondérées par la distance séparant les deux points (courbe bleue Fig. 2c).

Gestion de la sur-segmentation du fond : 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. Les courbes *splines* ne sont donc dessinées qu'à condition de ne pas intersecter le dessin de départ I_b ailleurs qu'aux extrémités et, de manière optionnelle, à condition de ne pas s'intersecter entre elles. De plus, un trait de fermeture n'est pas dessiné lorsque celui-ci ferait apparaître au moins une composante connexe de fond dont le nombre de pixels serait inférieur à un seuil $a_{\min}$ donné (cas de la région de couleur marron de la Fig. 2c si $a_{\min} = 100$ car la région compte 50 pixels). Un algorithme, linéaire en la taille du trait candidat, est utilisé pour faire ce test [8].

Connexion par des segments Après avoir tracé les *splines* jugées valides, des segments de droites issus de chacun des points d'intérêts sont éventuellement ajoutés (en vert sur la Fig. 2c). 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. Le tracé de ces traits est lui aussi conditionné à la vérification de plusieurs contraintes : une longueur maximum $s_{\max}$; la taille minimale des régions créées $a_{\min}$; et le nombre maximum de *splines* ou segments issus d'un même point d'intérêt.

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 à I .

1. Colorisation aléatoire des zones connexes : On génère un calque de colorisation C composé de régions de couleurs aléatoires, constantes par morceaux, formant un sur-partitionnement du dessin d'origine I (Fig. 4b). Le travail du préparateur d'aplats se résume par la suite à affecter une couleur crédible à chaque région de C (la même couleur pouvant être attribuée à des régions voisines (Fig. 4c)). La génération du calque C est réalisée par une labélisation en composante connexes de l'image binaire I_f avec l'algorithme rapide [9], pour les points $I_f(x, y) = 0$. Puis, ces labels sont propagés pour les points $I_f(x, y) = 1$, par un algorithme de propagation par queue de priorité [10], avec une priorité $P_{(x, y)} = -\min_{(p, q) \in \Omega} [\|(x, y) - (p, q)\| \mid I_f(p, q) = 1]$, qui aboutit à une reconstruction en *pelure d'oignon* des couleurs sur les contours résiduels. P se calcule efficacement par l'algorithme rapide [11] pour la fonction distance. On attribue enfin une couleur aléatoire à chaque label différent obtenu. La fermeture préalable des traits de dessin permet d'obtenir de nombreuses composantes connexes, et un sur-partitionnement judicieux du dessin d'origine I (par exemple la frontière ouverte front-cheveux en Fig. 4b).

2. Colorisation guidée par touches de couleurs : Cette deuxième technique reprend l'idée de l'extrapolation de touches de couleurs, placées par l'utilisateur, pour former une partition du dessin entier (Fig. 4d,e). Cette idée est celle que l'on retrouve dans [1, 2, 3, 4]. Là encore, la propagation des pixels labélisés par l'utilisateur se réalise par [10], avec une priorité $P_{(x,y)} = \min((I_f * G_{\sigma_1})_{(x,y)}, (I_f * G_{\sigma_2})_{(x,y)})$, où $I_f * G_{\sigma}$ est la convolution de I_f par un noyau gaussien 2d d'écart type σ . Deux échelles d'analyse locales/globales $\sigma_1 = 1$ et $\sigma_2 = \max(l, h)/100$ sont considérées pour l'estimation d'une fonction P continue. La prise en compte des contours fermés I_f pour la propagation des labels de couleurs joue un rôle prépondérant dans l'efficacité de la colorisation (Fig. 4f,g).

4 Conclusion

La Fig. 4 montre clairement que la fermeture des traits de dessin est une étape essentielle pour l'obtention de résultats de colorisation pertinents. Les différents paramètres de notre algorithme permettent en pratique de nous adapter à de nombreux types de dessins, et à des résolutions d'images différentes. L'utilisation de technique de labélisation/propagation de complexité quasi-linéaires assure à notre méthode un temps de calcul court, en comparaison des algorithmes de l'état de l'art (gain mesuré de 92% sur une image de résolution 2630×1200 , avec une implémentation CPU/C++ de [4]). Des résultats et démonstrations complémentaires sur notre méthode peuvent être trouvés dans le *pre-print* [8]. Pour finir, notons que dans un souci de reproductibilité scientifique, notre algorithme a été intégré dans le cadriciel libre de traitement d'image [12], permettant son utilisation libre par tout un chacun.

Références

- [1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Transactions on Graphics*, 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. Šykora, 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] S. Fourey and R. Malgouyres, "Normals estimation for digital surfaces based on convolutions," *Computers & Graphics*, vol. 33, no. 1, 2009.
- [6] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.
- [7] A. Rosenfeld, "Adjacency in digital pictures," *Information and Control*, vol. 26, no. 1, pp. 24–33, 1974.
- [8] S. Fourey, D. Tschumperlé, and D. Revoy, "Un algorithme semi-guidé performant de colorisation en aplats pour le dessin au trait." hal-01490269, Mar. 2017. Technical report.
- [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] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time," in *Int. Symp. on Math. Morpho. and its Appl. to Image and Signal Proc.*, 2000.
- [12] D. Tschumperlé, "G'MIC: GREYC's Magic for Image Computing: A full-featured open-source framework for image processing." <http://gmic.eu/>, 2008–2017.

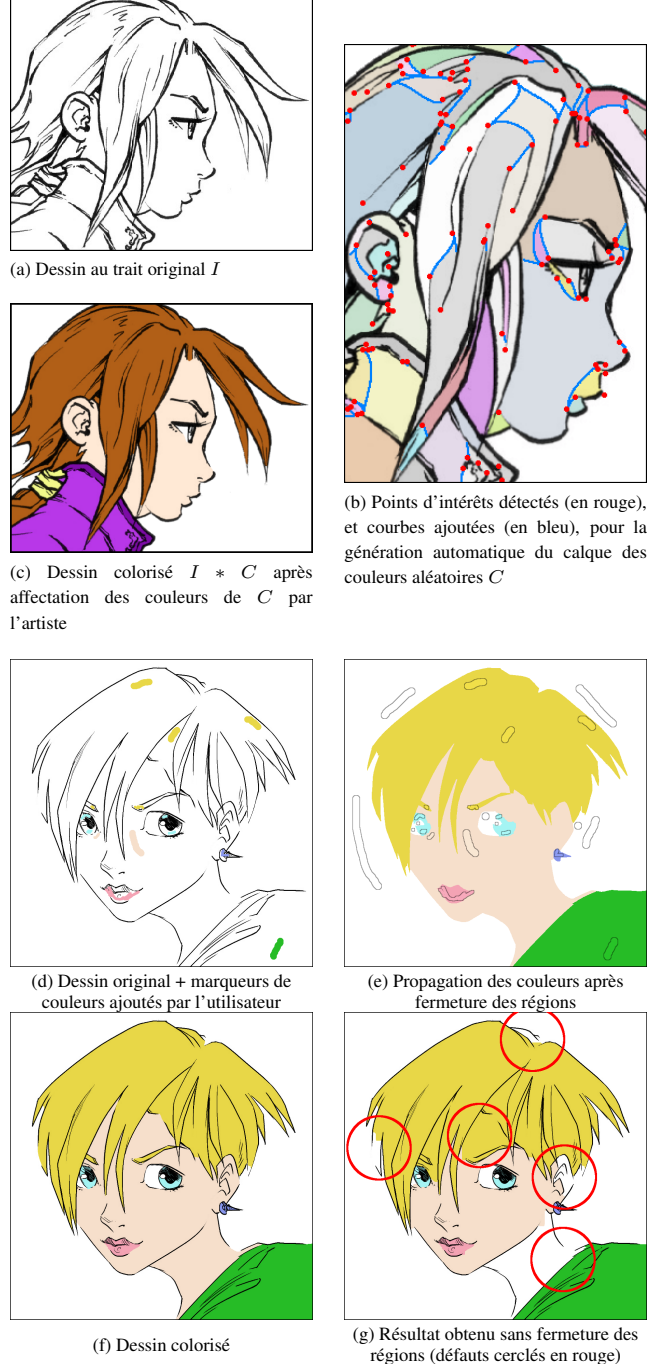


FIG. 4: Illustration des deux méthodes de colorisation proposées. **Méthode 1 (a,b,c) :** Colorisation par affectation manuelle de couleurs sur un calque de couleurs aléatoires sur-segmenté. **Méthode 2 (d,e,f,g) :** Colorisation par propagation de marqueurs colorés placés par l'utilisateur.