G’MIC 2.0 : un second souffle pour le traitement d’images libre

Posté par **David Tschumperlé** (**site web personnel**) le 02/06/17 à 10:38. Édité par **7 contributeurs**. Modéré par **bubar**. **Licence CC By-SA.**

Étiquettes : **gimp** , **gtk** , **qt** , **g'mic** , **traitement_d'images** **+ Étiqueter**

L’équipe *IMAGE* du laboratoire ***GREYC*** de Caen (*UMR CNRS 6072*) est heureuse de vous annoncer la sortie d’une nouvelle version majeure (numérotée 2.0) de son projet ***G’MIC***, un cadriciel *libre*, générique et extensible pour le **traitement des images** [W].
Nous exposons ici les avancées principales réalisées sur cet ensemble d’interfaces logicielles, depuis notre **dernière dépêche**. Les nouveautés présentées ici englobent le travail réalisé ces douze derniers mois (versions 2.0.0 et 1.7.*x*, pour *x* variant de 2 à 9).

Sommaire

N. D. A. : Cliquez sur les images de la dépêche pour en visualiser des versions à meilleure résolution.

1. Contexte et vue d’ensemble

G’MIC est un projet libre ayant vu le jour en août 2008, dans l’équipe **IMAGE** du **GREYC**.
Cette équipe française, composée de chercheurs et d’enseignants-chercheurs, est spécialisée dans les domaines de l’algorithmique et des mathématiques du traitement d’images. *G’MIC* est distribué sous licence **CeCILL** (compatible GPL) pour différentes plates-formes (GNU/Linux, macOS et Windows). Il fournit un ensemble d’interfaces utilisateur variées pour la manipulation de données images *génériques*, à savoir des images ou des séquences d’images hyperspectrales 2D ou 3D à valeurs flottantes (ce qui inclut de fait les images couleurs « classiques »).

Fig. 1.1 : Logo du projet G'MIC, cadriciel libre pour le traitement d'image, et sa mascotte Gmicky.

La popularité de G'MIC vient en majeure partie du **greffon** qu'il propose pour le logiciel **GIMP**, depuis 2009. À ce jour, celui-ci dispose de plus de 480 filtres et effets différents à appliquer sur vos images, et permet donc d'enrichir considérablement les filtres de traitement livrés par défaut avec GIMP.

G'MIC fournit également une interface en **ligne de commande**, relativement puissante, qui peut s'utiliser de manière autonome, et qui est complémentaire aux outils en ligne de commande proposés par **ImageMagick** ou **GraphicsMagick**. Il existe aussi un service Web ***G'MIC Online***, permettant d'appliquer des effets sur vos images directement à partir d'un navigateur. D'autres interfaces plus confidentielles basées sur G'MIC existent : l'interface de diffusion vidéo ***ZArt***, un greffon pour **Krita**, un sous-ensemble de filtres pour **Photoflow**, **Blender** ou encore **Natron**… Toutes ces interfaces se basent sur les bibliothèques *C++* ***CImg*** et ***libgmic*** qui sont portables, compatibles multi-fil d'exécution (*thread-safe* et *multi-thread*), via l'utilisation d'**OpenMP**.

À l'heure actuelle, G'MIC possède plus de **950 fonctions** différentes de traitement, toutes paramétrables, pour une bibliothèque de seulement 6,5 Mio, représentant un peu plus de 180 000 lignes de code source. Les fonctionnalités proposées couvrent un large spectre du traitement d'images, en proposant des algorithmes pour la manipulation géométrique, les changements colorimétriques, le filtrage d'image (débruitage, rehaussement de détails par méthodes spectrales, variationnelles, non locales…), l'estimation de mouvement et le recalage, l'affichage de primitives (2D et objets 3D maillés), la détection de contours et la segmentation, le rendu artistique, etc. C'est donc un outil très générique aux usages variés et très utile, d'une part pour convertir, visualiser et explorer des données images, et d'autre part, pour construire des *pipelines* élaborés et des algorithmes personnalisés de traitements d'images (voir **les diapositives** de présentation du projet pour davantage de détails).

2. Une nouvelle interface polyvalente, basée sur Qt

La nouveauté essentielle de cette version 2.0 concerne le code du greffon, qui a été complètement repensé et réimplémenté à partir de zéro. Ce module, ***G'MIC-Qt***, développé par **Sébastien** (enseignant-chercheur — expérimenté — de l'équipe), propose ainsi une refonte complète en Qt de l'interface du greffon, en étant le plus indépendant possible de l'interface logicielle et des *widgets* de GIMP.

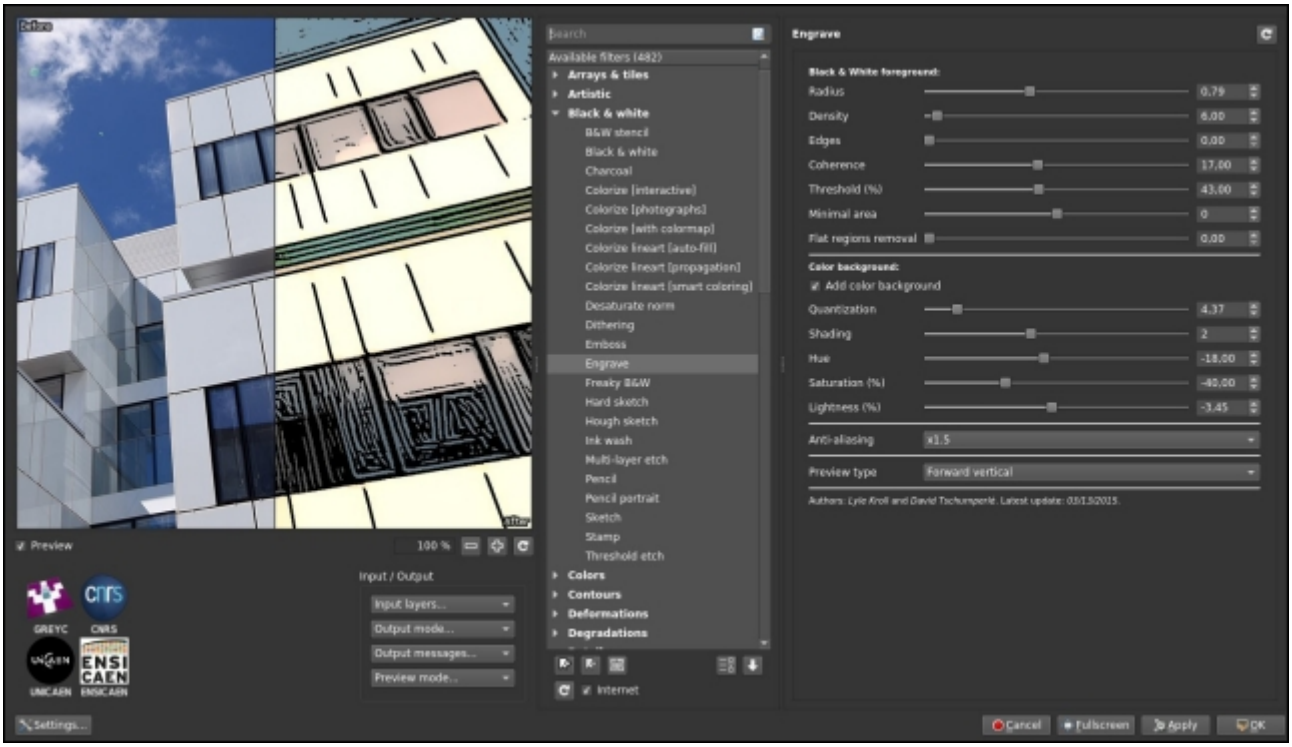

Fig. 2.1 : Aperçu de la version 2.0 du greffon G'MIC-Qt pour GIMP.

Cela a plusieurs conséquences intéressantes :

- Le greffon utilise ses propres *widgets* (en Qt) et permet d'avoir une interface plus flexible et personnalisable qu'avec les *widgets* GTK proposés par défaut pour les greffons dans l'API de GIMP : la fenêtre de prévisualisation des filtres est par exemple redimensionnable à volonté, gère le zoom à la molette de la souris, et peut être localisée à gauche ou à droite de l'interface. Un moteur de recherche de filtres par mots-clés a été ajouté, ainsi que la possibilité de choisir entre un thème clair ou sombre. La gestion des filtres favoris a été améliorée et l'interface propose même un mode d'édition de visibilité des filtres pour une personnalisation maximale.
- Le greffon définit aussi sa propre *API*, pour faciliter son intégration dans des logiciels tiers autres que *GIMP*. Pour le développeur, il s'agit en pratique d'écrire un unique fichier `host_software.cpp` implémentant les fonctions de l'API pour faire le lien entre le greffon et l'application hôte. Actuellement, le fichier `host_gimp.cpp` réalise ce travail pour le logiciel hôte GIMP. Mais il existe désormais aussi une version *stand-alone* (fichier `host_none.cpp`) qui propose de lancer l'interface graphique Qt du greffon en mode solo, depuis la ligne de commande (commande `gmic_qt`).
- **Boudewijn Rempt**, gestionnaire et développeur du projet **Krita** a également débuté l'écriture du fichier `host_krita.cpp` pour la communication de ce greffon « nouvelle génération » avec le logiciel de peinture numérique Krita. Il s'agit à terme de remplacer le greffon G'MIC actuellement distribué avec *Krita*, qui se fait vieillissant et qui pose des problèmes de maintenance pour les développeurs.

Minimiser l'effort d'intégration pour les développeurs, partager au maximum le code du greffon G'MIC_ entre différentes applications, et proposer une interface d'utilisation la plus confortable possible ont été les objectifs principaux de cette refonte complète. Comme vous vous en doutez, cette réécriture a demandé un travail long et soutenu, et nous ne pouvons qu'espérer que cela suscitera de l'intérêt chez d'autres développeurs de logiciels, où disposer d'une base conséquente de filtres de traitement d'images peut être utile (à quand un `host_blender.cpp` ? On peut rêver !). L'animation ci-dessous illustre quelques-unes des fonctionnalités offertes par cette nouvelle interface basée sur Qt.

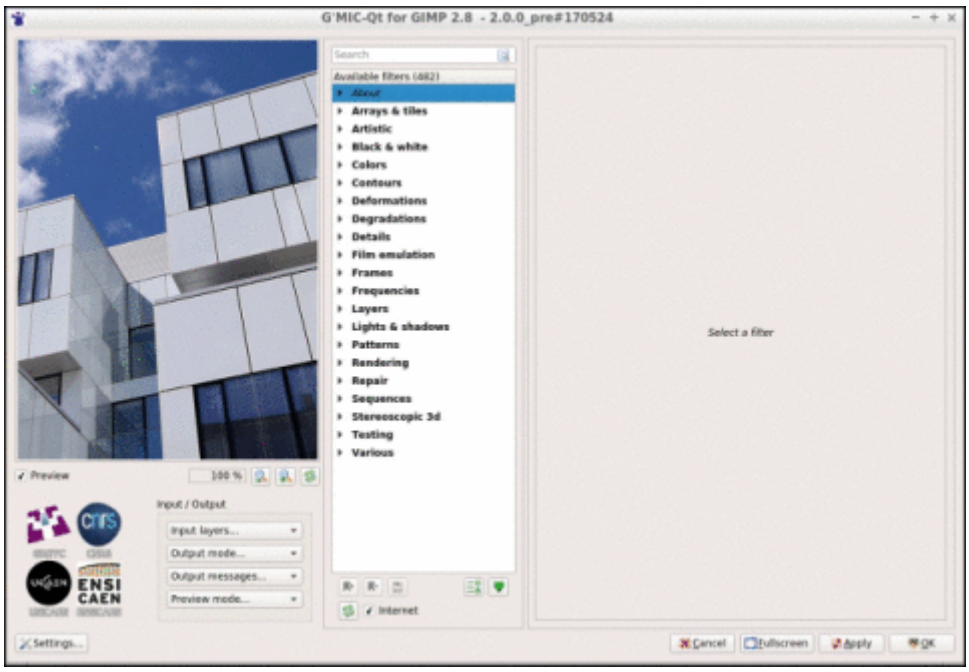

Fig. 2.2 : La nouvelle interface G'MIC-Qt en action.

À noter que le code de l'ancien greffon utilisant **GTK** a quand même été mis à jour pour fonctionner avec la version 2.0 de G'MIC, mais possède moins de fonctionnalités et n'évoluera sans doute plus dans le futur, contrairement à la version Qt.

3. Faciliter la vie des dessinateurs...

Mais une des raisons d'être de G'MIC, c'est aussi de proposer toujours plus de filtres et de fonctions pour traiter les images. Et ça tombe bien, car dans ce domaine, nous n'avons pas non plus relâché nos efforts, malgré le nombre de filtres conséquent déjà disponible dans les versions précédentes !

Une des grandes nouveautés de cette version concerne l'apparition et l'amélioration de filtres pour l'aide à la colorisation de dessins au trait (*line-arts*). Nous avons en effet eu la chance de recevoir au laboratoire pendant quelques jours l'artiste **David Revoy**, bien connu des amateurs d'art et de logiciels libres de par ses contributions multiples dans ces domaines (en particulier, son *webcomic* ***Pepper & Carrot***, à lire absolument !). En collaboration avec David, nous avons travaillé sur la conception d'un filtre original de colorisation automatique de dessins au trait, nommé ***Smart Coloring***.

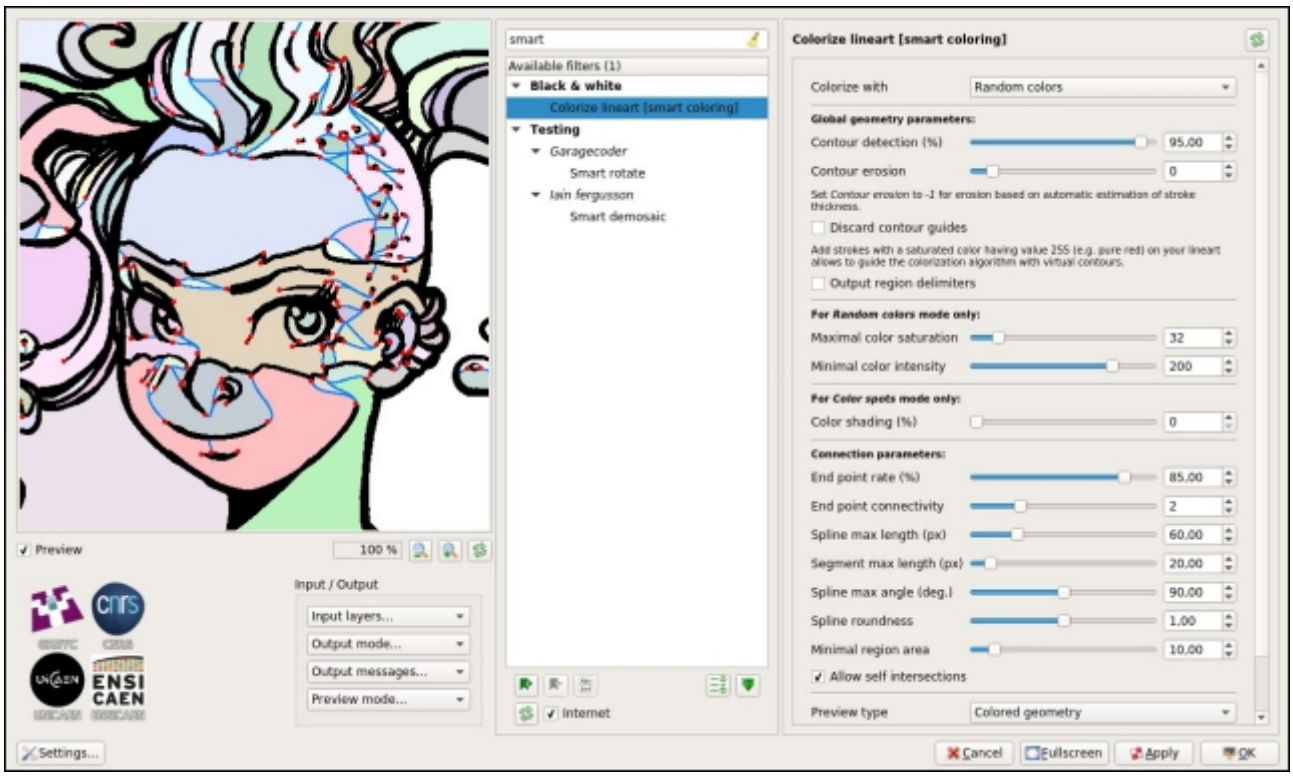

Fig. 3.1 : Utilisation du filtre « Colorize lineart [smart coloring] » de G'MIC.

Dans le domaine de la bande dessinée, la colorisation d'un dessin au trait se réalise en effet en deux étapes successives : Le dessin d'origine en niveaux de gris (*fig. 3.2.[1]*) est d'abord pré-colorisé en aplats, c.-à-d. en attribuant une couleur unique à chaque région ou objet distinct du dessin (*fig. 3.2.[3]*). Dans un second temps, le coloriste retravaille ce pré-coloriage, ajoutant ombres, lumières et modifiant l'ambiance colorimétrique, pour obtenir le résultat de colorisation final (*fig. 3.2.[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. Ce calque est fusionné avec le *line-art* d'origine pour obtenir le rendu colorisé du dessin en aplats (fusion des calques en mode *multiplication*, typiquement).

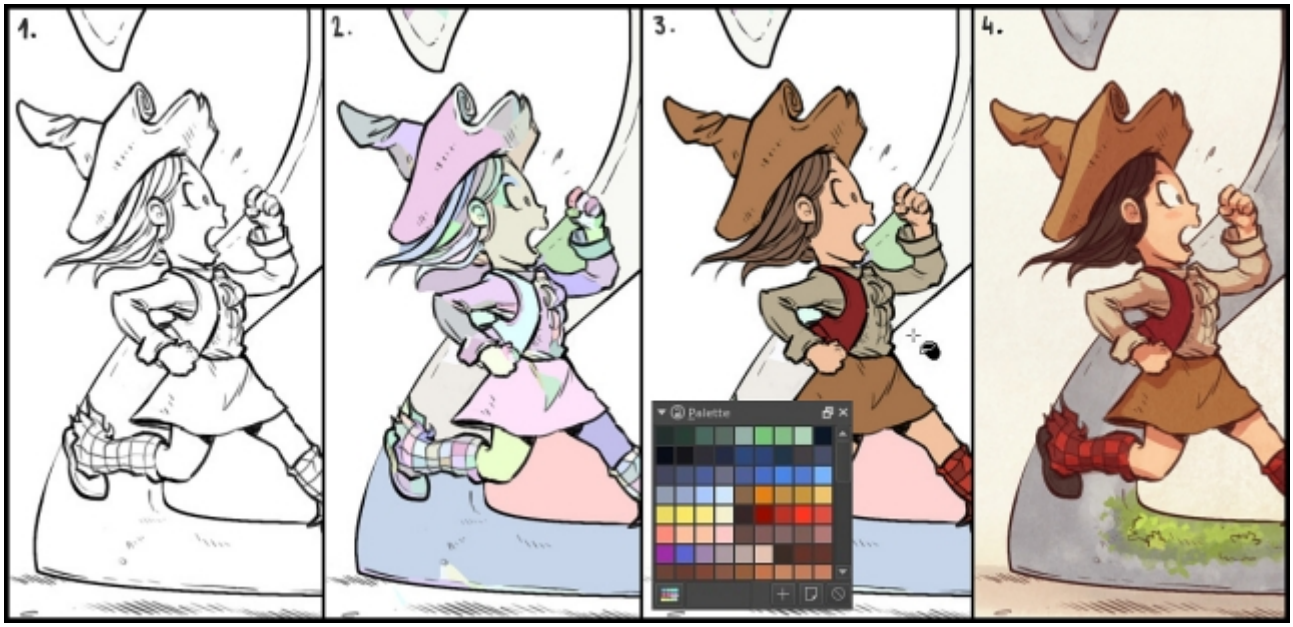

Fig. 3.2 : Étapes d'un processus de colorisation d'un dessin au trait (source : David Revoy).

Les artistes l'avouent eux-mêmes : 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 (*bucket fill*), même les plus évolués, gèrent mal les discontinuités dans les traits de dessin (*fig. 3.3.a*), et encore plus mal les traits lissés (*anti-crénelés*). Il est alors courant que l'artiste effectue ses aplats en peignant manuellement ses couleurs avec une brosse sur un calque séparé (*fig. 3.3.b*), avec tous les problèmes de précision (notamment aux abords des traits) que cela suppose (*fig. 3.3.c*, et ***ce lien*** pour plus de détails).

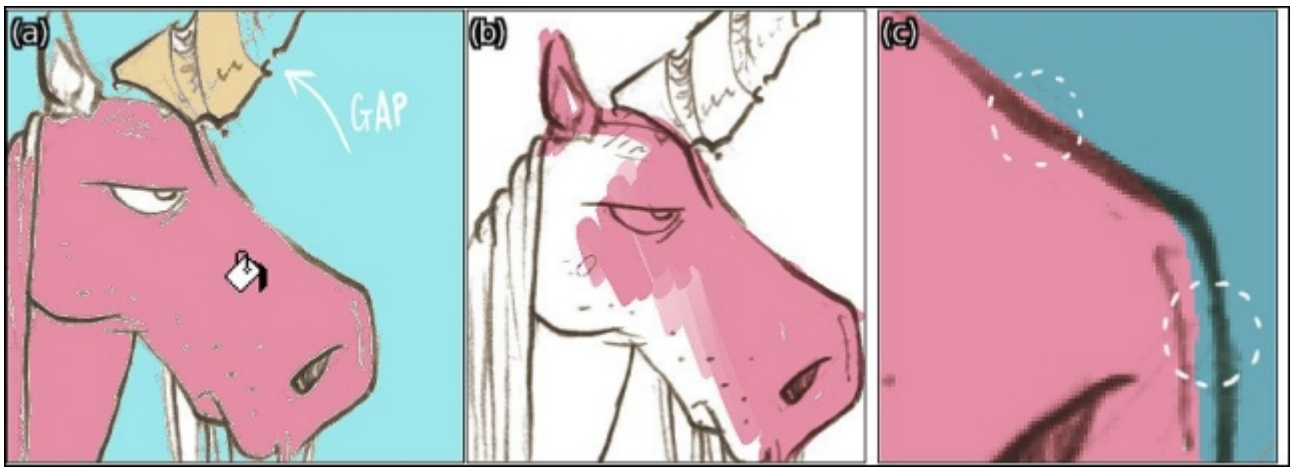

Fig. 3.3 : Problèmes classiques rencontrés lors de la colorisation en aplats (source : David Revoy).

Il peut même arriver que l'artiste décide de contraindre explicitement son style de dessin, en utilisant par exemple des lignes crénelées en résolution supérieure (plutôt que des lignes anti-crénelées), et/ou en se forçant à ne tracer que des traits sans « trous », pour faciliter le travail de colorisation du préparateur d'aplats.

Le filtre *Smart Coloring* que nous avons développé dans la version 2.0 de G'MIC permet de pré-coloriser un dessin au trait donné en entrée, d'une part à partir d'une analyse de la géométrie locale des traits de crayon (normales et courbures) et, d'autre part, par leur complétion automatique par des courbes ***splines*** [W]. Cela autorise une fermeture (virtuelle) et un remplissage d'objets pouvant contenir des discontinuités dans les tracés de leurs contours. Ce filtre a de plus l'avantage d'être rapide en temps de calcul et donne des résultats de colorisation de qualité similaire à des techniques d'optimisation plus coûteuses, utilisées dans certains logiciels propriétaires. L'algorithme que nous avons élaboré gère sans problème les traits anti-crénelés, et propose deux modes de colorisation en aplats : par couleurs aléatoires (*fig. 3.2.[2]* et *fig. 3.4*) ou guidée par des marqueurs de couleur placés au préalable par l'utilisateur (*fig. 3.5*).

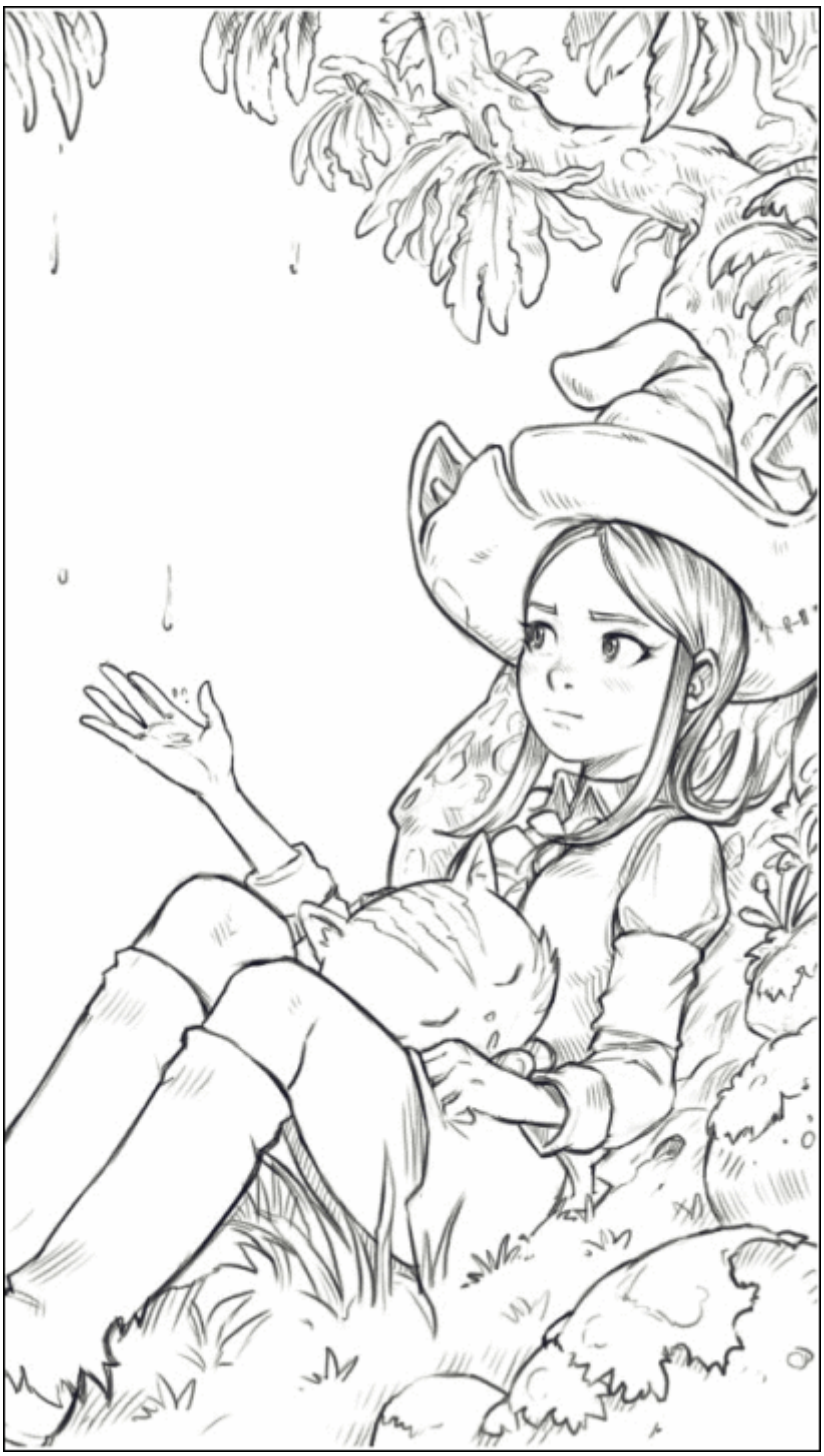

Fig. 3.4 : Utilisation du filtre « Smart Coloring » en mode couleurs aléatoires, pour la colorisation d'un dessin au trait (source : David Revoy).

En mode *couleurs aléatoires*, le filtre génère un calque d'aplats qu'il est très facile de recoloriser par la suite avec de « bonnes » couleurs. Ce calque contient en effet uniquement des régions de couleurs constantes, et l'outil de remplissage classique (*bucket fill*) est ici efficace pour réassigner rapidement une couleur cohérente à chaque région existante colorisée aléatoirement par l'algorithme.
En mode *marqueurs de couleurs*, les couleurs placées par l'utilisateur sont extrapolées de façon à respecter au mieux la géométrie du dessin, notamment avec la prise en compte des discontinuités dans les traits de crayon, comme on peut le voir très clairement sur l'exemple ci-dessous :

Fig. 3.5 : Utilisation du filtre « Smart Coloring » en mode guidage par marqueurs de couleurs, pour la colorisation d'un dessin au trait (source : David Revoy).

Cet algorithme innovant de colorisation en aplats a fait l'objet d'une pré-publication sur *HAL* : ***Un algorithme semi-guidé performant de colorisation en aplats pour le dessin au trait***. Les plus curieux y trouveront tous les détails techniques de la méthode utilisée.
Les discussions récurrentes que nous avons eues avec David Revoy sur l'élaboration de ce filtre nous ont permis d'améliorer l'algorithme petit à petit, jusqu'à ce qu'il devienne réellement utilisable en production. Cette méthode a pu être ainsi utilisée avec succès, et donc validée, pour la pré-colorisation de tout **l'épisode 22** du *webcomic Pepper & Carrot*.

Les plus avisés pourraient nous dire que G'MIC possédait déjà un **filtre de colorisation de dessins** ! C'est vrai, mais celui-ci ne gérait malheureusement pas très bien les dessins avec des traits discontinus (tels que celui présenté en *fig. 3.5*), et pouvait alors nécessiter de la part de l'utilisateur le placement d'un grand nombre de marqueurs couleur pour guider correctement l'algorithme. La performance de ce nouvel algorithme est en pratique bien supérieure.

Et puisque ce nouveau filtre d'aide à la colorisation ne voit pas d'objection aux traits anti-crénelés, pourquoi s'en priver ? Justement, un autre filtre *Repair / Smooth [antialias]* a également fait son apparition pour ajouter de l'anti-crénelage à des traits de dessin qui seraient à l'origine trop crénelés.

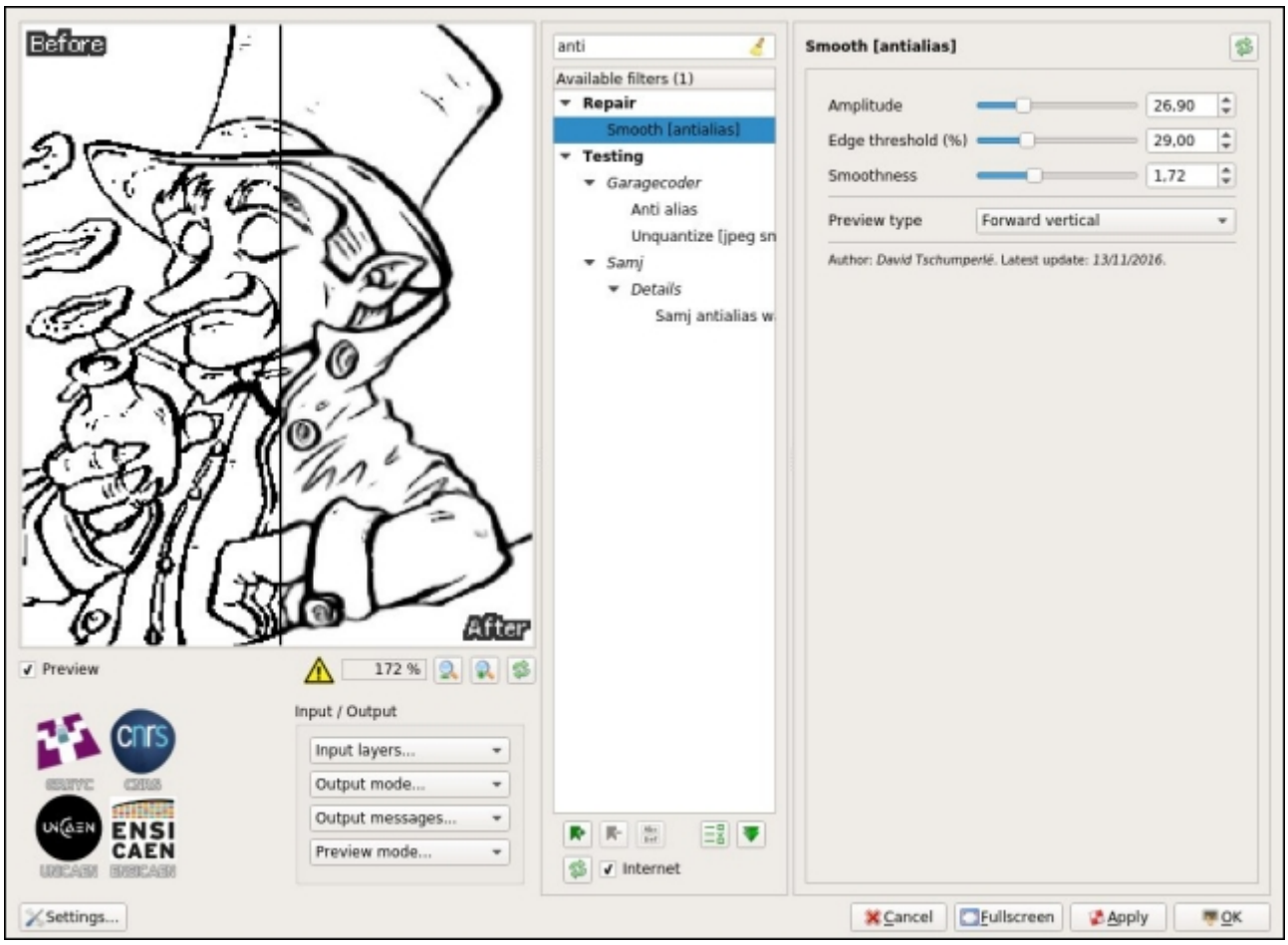

Fig. 3.6 : Le filtre « Smooth [antialias] » peut supprimer le crénelage éventuel de traits de dessins (source : David Revoy).

4. ...Sans oublier les photographes !

« *Coloriser des dessins, c'est bien gentil, mais mes photos à moi sont déjà en couleurs !* », nous fait gentiment remarquer le photographe impatient. Pas de souci ! De nombreux filtres relatifs à la transformation et l'amélioration de photos ont également fait leur apparition dans G'MIC 2.0. Faisons un petit tour du propriétaire.

4.1. CLUT et transformations colorimétriques

Les **CLUT** (*Color Lookup Tables*) sont des fonctions de transformations colorimétriques définies dans le cube RVB (rouge, vert et bleu) : pour chaque couleur source d'une image *Imgs* définie par un triplet *(Rs, Vs, Bs)*, une CLUT lui associe une nouvelle couleur *(Rd, Vd, Bd)* que l'on reporte dans une image destination *Imgd* aux mêmes positions. Ces fonctions de transformation pouvant être vraiment quelconques, on peut obtenir des effets très différents suivant les CLUT utilisées, et les photographes en sont donc généralement friands (d'autant que ces CLUT sont un bon moyen de simuler le rendu couleur de certains vieux films argentiques).

En pratique, on stocke une CLUT comme une image volumique couleur (éventuellement « déroulée » suivant l'axe $z = B$ pour en obtenir **une version 2D**), ce qui peut vite devenir encombrant lorsque l'on doit gérer plusieurs centaines de CLUT. Heureusement, G'MIC possède un algorithme de compression spécifique de CLUT assez efficace (déjà mentionné dans **une précédente dépêche *LinuxFr.org***), qui a été amélioré au fil des versions. Et c'est donc en toute décontraction que nous avons pu ajouter plus de **60** nouvelles transformations colorimétriques basées CLUT dans G'MIC, pour un total de **359** CLUT utilisables, toutes stockées dans un fichier de données qui pèse à peine 1,2 Mio ! Remercions en passant **Marc Roovers** et **Stuart Sowerby** pour leurs contributions à ces nouvelles transformations.

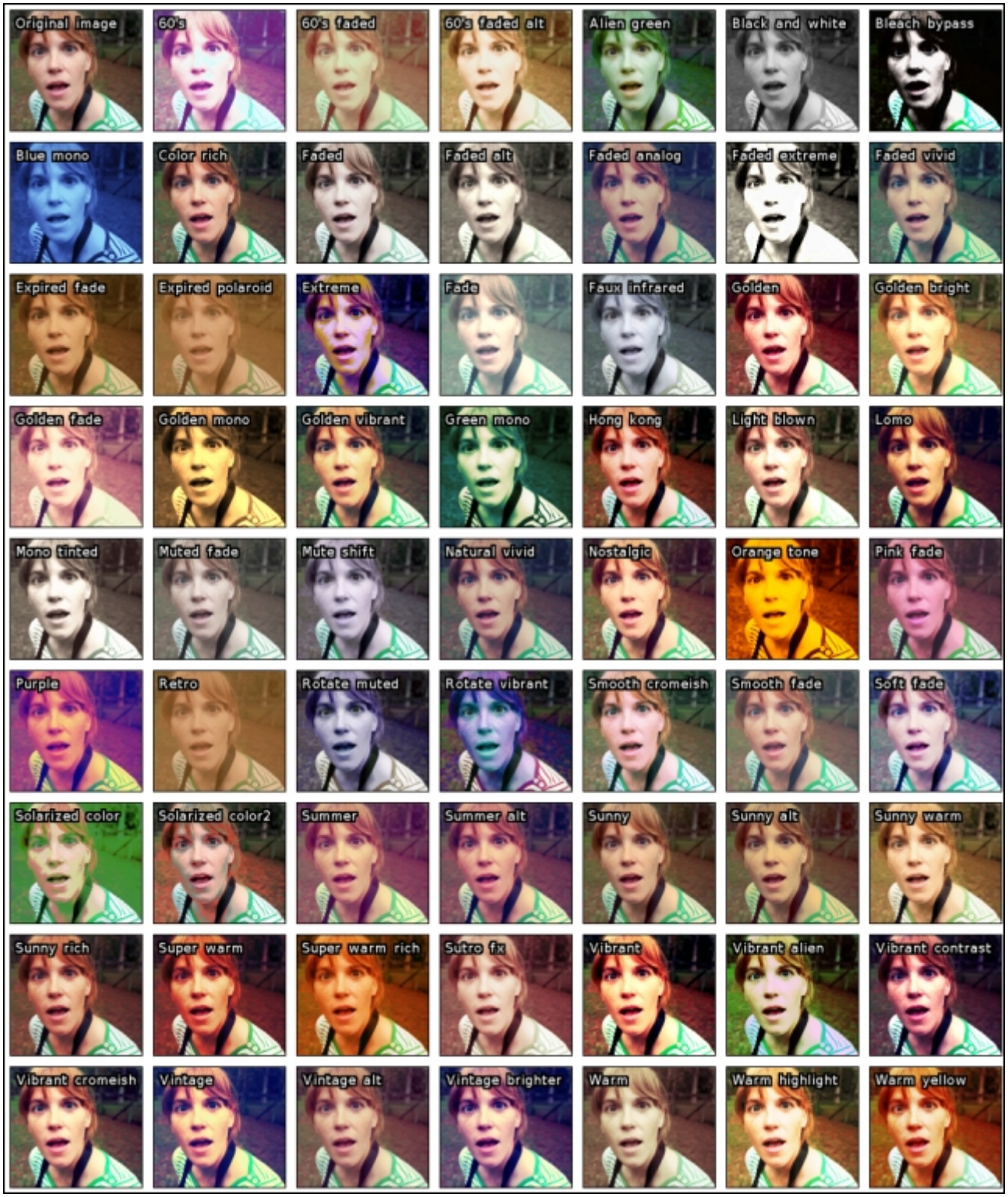

Fig. 4.1.1 : Quelques-unes des nouvelles transformations colorimétriques disponibles dans G'MIC (source : Pat David).

Mais que faire si nous disposons déjà de nos propres fichiers CLUT et que nous voulons les utiliser dans GIMP ? Aucun problème ! Le filtre tout frais *Film emulation / User-defined* vous permet d'appliquer de telles transformations, avec même une prise en charge des fichiers à extension `.cube` (**format de fichiers CLUT** proposé par Adobe et encodé en ASCII `o_0` !).

Et les plus exigeants, qui ne trouveraient pas leur bonheur dans les CLUT déjà présentes, peuvent dorénavant construire leur propre CLUT personnalisée, grâce au filtre *Colors/Customize CLUT* disponible dans le greffon. Ce filtre, très polyvalent, permet de placer des points clefs colorés dans le cube RVB, points qui seront interpolés en 3D par la suite (suivant une **triangulation de Delaunay**[W]), pour reconstruire une CLUT complète, c.-à-d. une fonction dense dans *RGB*. Toutes les fantaisies sont alors permises, comme dans l'exemple ci-dessous, où le filtre est utilisé pour changer l'ambiance colorimétrique d'un paysage, en modifiant principalement la couleur du ciel. La CLUT ainsi synthétisée peut être, bien sûr, sauvegardée sous forme de fichier et réutilisée plus tard pour

d’autres photographies, ou même dans d’autres logiciels prenant en charge ce type de transformations colorimétriques (**RawTherapee** ou **Darktable**, par exemple).

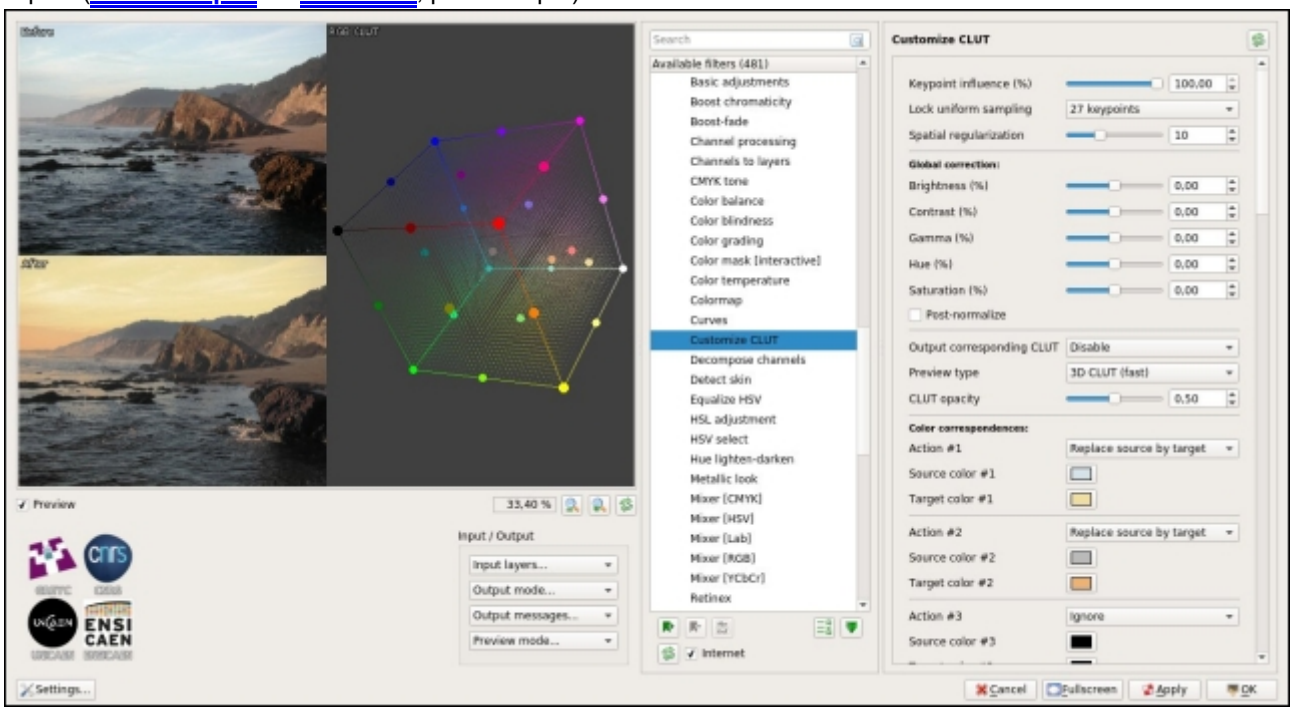

Fig. 4.1.2 : Création d’une transformation colorimétrique personnalisée avec le filtre « Customize CLUT ».

Fig. 4.1.3 : Résultat de la transformation colorimétrique personnalisée appliquée sur un paysage.

Toujours pour rester dans des modifications relatives aux couleurs, notons aussi l’apparition du filtre *Colors/Retro fade* qui créé un rendu « rétro » d’une image avec du grain généré par moyennages successifs de quantifications aléatoires d’une image d’entrée.

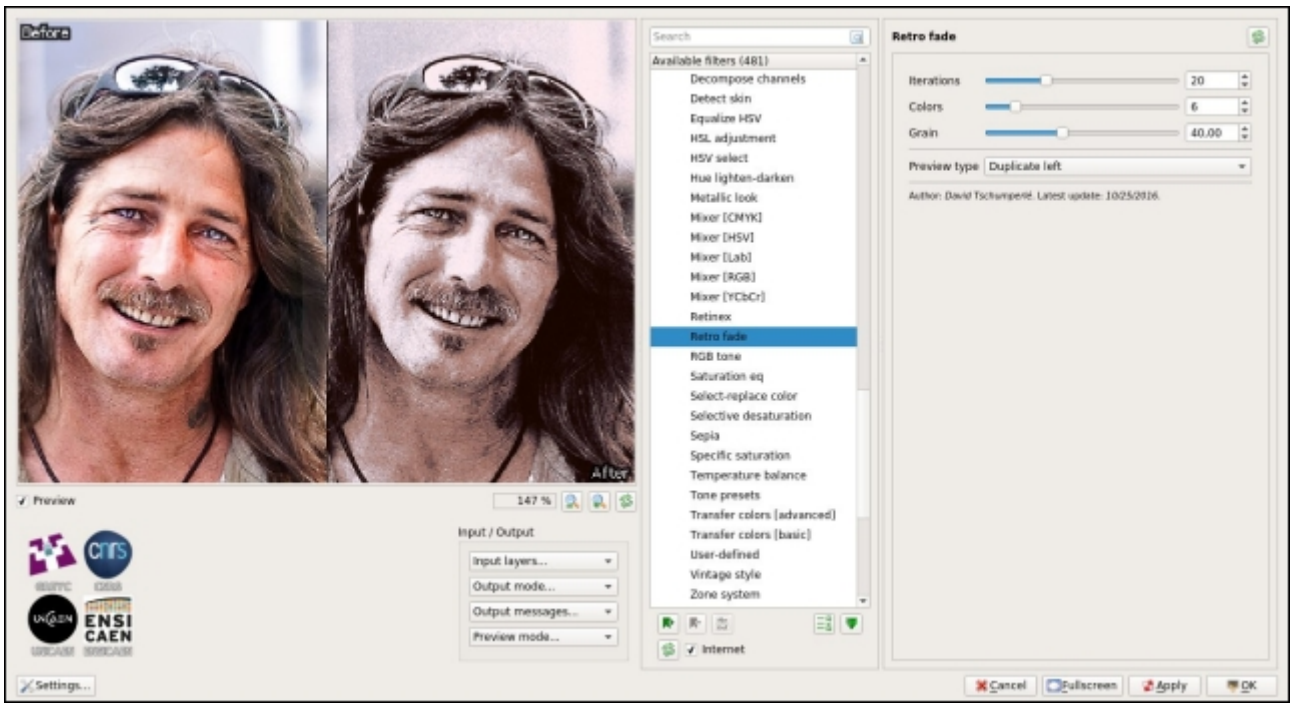

Fig. 4.1.4 : Le filtre « Retro fade » du greffon G'MIC.

4.2. Faire ressortir les détails

De nombreux photographes cherchent des moyens de traiter leurs photographies numériques de façon à faire ressortir les moindres détails de leurs images, parfois même jusqu'à l'exagération. On en croise notamment sur ***PixIs.us***, site récent spécialisé dans les techniques de photographie utilisant uniquement des projets libres. Et c'est en collaboration avec ceux-ci, que plusieurs nouveaux filtres de rehaussement de détails et de contraste ont été ajoutés dans G'MIC. On peut citer notamment les filtres *Artistic/Illustration look* et *Artistic/Highlight bloom*, ré-implémentations directes des tutoriels et des scripts rédigés par **Sébastien Guyader**, ainsi que le filtre *Light&Shadows / Pop shadows* suggéré par **Morgan Hardwood**. Être immergé dans une telle communauté de photographes donne sans cesse des opportunités de création de nouveaux effets !

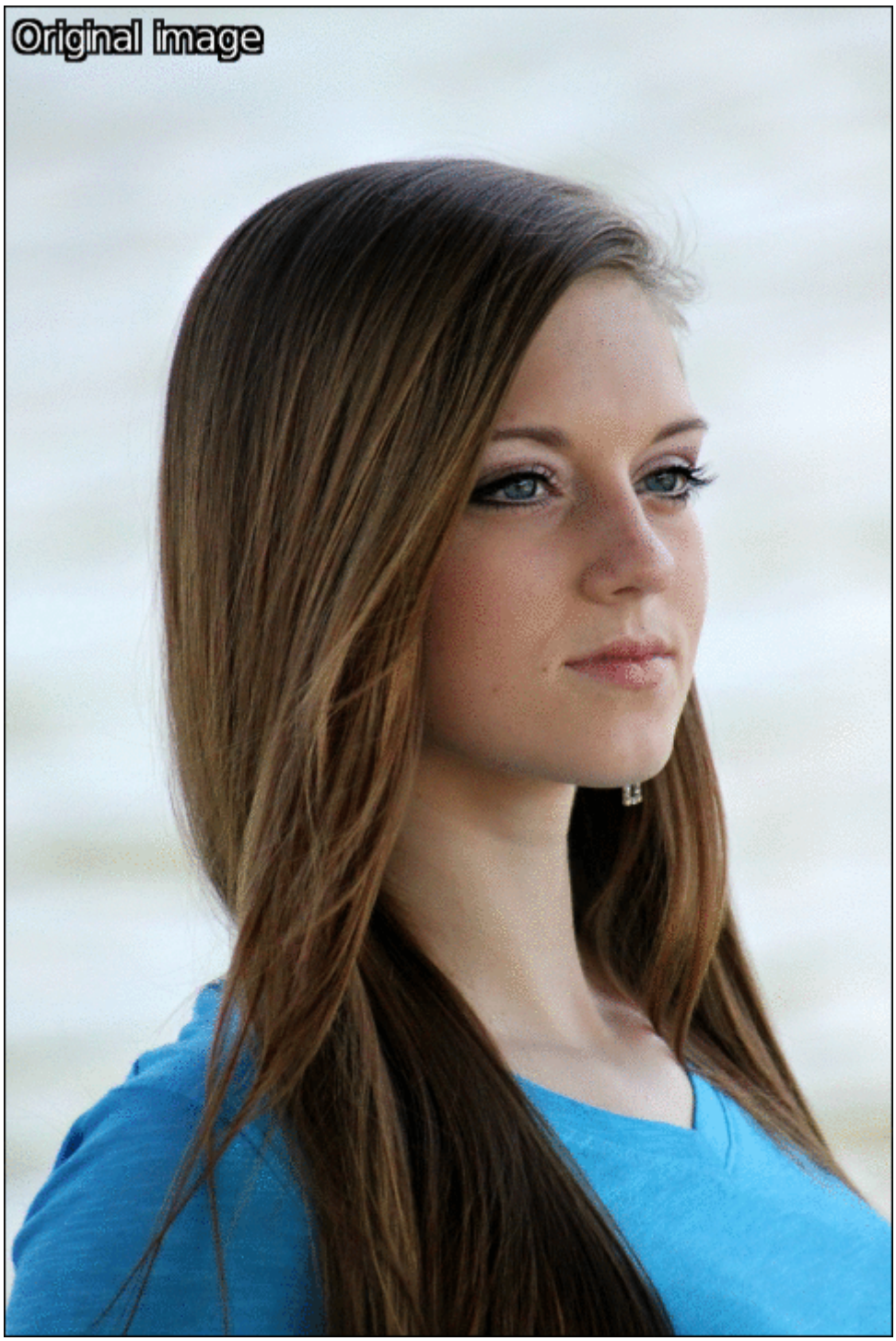

Fig. 4.2.1 : Filtres « Illustration look » et « Highlight bloom » appliquées sur une image de portrait.

Dans la même veine, G'MIC se dote de sa propre implémentation de l'algorithme de **Retinex multi-échelle**, dont une version était **déjà présente** dans GIMP, mais qui se trouve ici enrichie avec des possibilités de contrôle supplémentaires, pour améliorer la cohérence de la luminosité dans les images.

Fig. 4.2.2 : Filtre « Retinex » pour uniformiser la luminosité dans une image.

Notre ami et grand contributeur de G'MIC, **Jérome Boulanger**, a également implémenté et ajouté un algorithme d'atténuation de l'effet de brouillard dans les images (*Dehazing*), basé sur la méthode du ***Dark Channel Prior***. Le réglage des paramètres de ce filtre n'est pas toujours aisé, mais il donne parfois des résultats très intéressants.

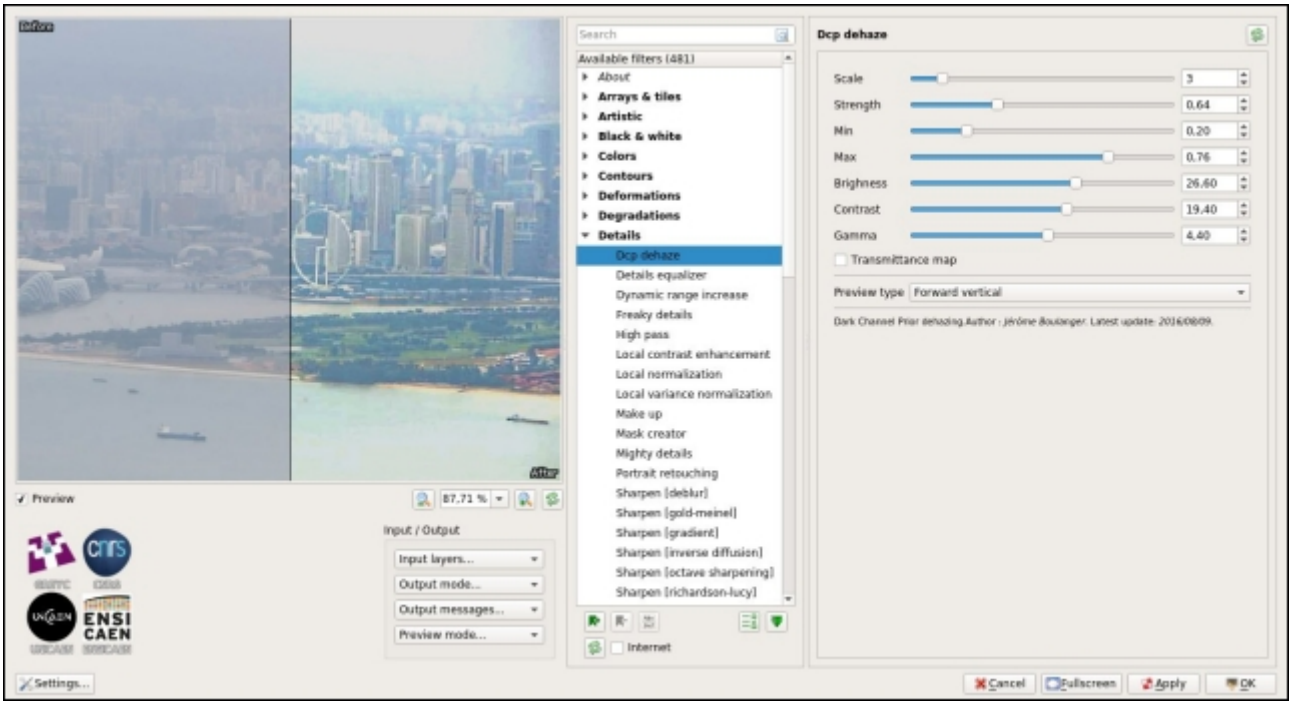

Fig. 4.2.3 : Filtre « DCP Dehaze » pour l'atténuation de l'effet de brouillard dans une image.

Pour terminer cette sous-section, notons également l'implémentation dans G'MIC de l'algorithme de ***Rolling Guidance***, qui permet de simplifier des images et qui est devenu une brique de base utilisée dans de nombreux filtres récemment ajoutés. C'est le cas notamment pour cette méthode épatante de rehaussement de contours (***Sharpening*** [W]), que vous pourrez trouver dans *Details / Sharpen [texture]*, et qui se réalise en deux étapes successives. D'abord l'image est séparée en une composante *texture* + une composante *couleur*, puis on rehausse les détails de la composante *texture* uniquement, avant de recomposer l'image. Cette façon de procéder permet de bien faire ressortir tous les petits détails de l'image, tout en minimisant la création de halos disgracieux près des contours, problème récurent rencontré avec la plupart des méthodes de rehaussement de détails plus classiques (tel que le bien connu ***Unsharp Mask*** [W]).

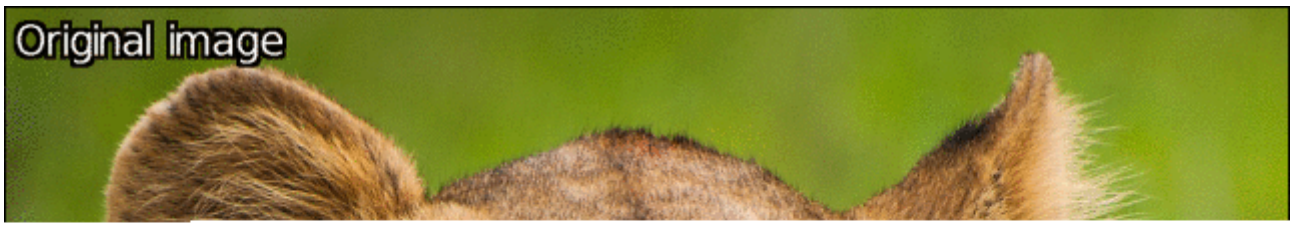

Fig. 4.2.4 : Le filtre « Sharpen [texture] » illustré pour deux amplitudes différentes de rehaussement de détails.

4.3. Masquage par sélection de couleurs

Il existe une multitude de techniques de retouche numérique de photographies se basant sur la création au préalable d'un ou plusieurs « masques », c.-à-d. des zones spécifiques de l'image qui vont recevoir des traitements différenciés. Par exemple, la technique très courante des **masques de luminosité** permet de traiter différemment les zones d'ombre et de lumière dans une photographie. G'MIC 2.0 se dote d'un nouveau filtre *Colors / Color mask [interactive]*, implémentant un algorithme relativement rusé (quoique coûteux en temps de calcul) d'aide à la création de masques. Ce filtre demande à l'utilisateur de passer sa souris au-dessus de quelques points représentatifs des régions qu'il souhaite isoler. L'algorithme apprend en temps réel les couleurs ou les luminosités correspondantes et en déduit l'ensemble des pixels qu'il doit conserver dans l'image entière (par **analyse en composante principale** [W] sur les échantillons RVB), pour la construction rapide d'un masque global.

Une fois le masque généré par ce filtre, l'utilisateur peut modifier les pixels correspondants avec tout type de traitement. L'exemple ci-dessous illustre l'utilisation de ce filtre pour modifier radicalement la couleur d'une voiture (l'opération se réalise en moins d'une minute trente ! Voir **la vidéo** correspondante). Une **autre vidéo** expose une technique identique pour modifier la couleur du ciel dans un paysage.

Fig. 4.3.1 : Modification de la couleur d'une voiture à l'aide du filtre « Color mask [interactive] ».

5. Et pour les autres...

Les dessinateurs et les photographes étant maintenant rassurés, passons à d'autres types de traitements, récemment ajoutés à G'MIC, à usage plus exotique, mais qui restent néanmoins intéressants !

5.1. Moyenne et médiane d'une série d'images

Vous êtes vous déjà demandé comment calculer facilement l'image moyenne ou médiane d'une série d'images d'entrée ? Le photographe **Pat David**, *aficionado* du Libre (il est le créateur du site ***PixIs.us***), s'est, quant à lui, souvent posé la question. Tout d'abord pour tenter de débruiter ses images, **en combinant plusieurs clichés d'une même scène**. Puis, **pour simuler un temps d'exposition plus long** par moyennage de photographies prises à des instants successifs. Et enfin, en calculant des moyennes d'images diverses et variées à des fins artistiques (par exemple, des trames de **clips vidéo musicaux**, l'ensemble des **couvertures de *Playboy***, ou encore des **portraits de célébrités**).

C'est donc avec sa bénédiction que nous avons ajouté les commandes `-median_files`, `-median_videos`, `-average_files` et `-average_videos` pour réaliser toutes ces choses très simplement, à partir de l'interface en ligne de commande `gmic` de G'MIC. L'exemple ci-dessous illustre le résultat de ces commandes appliquées sur une portion de la vidéo de ***Big Buck Bunny***. Nous avons tout simplement invoqué les commandes suivantes depuis Bash :

```
$ gmic -average_video bigbuckbunny.mp4 -normalize 0,255 -o average.jpg
$ gmic -median_video bigbuckbunny.mp4 -normalize 0,255 -o median.jpg
```

Fig. 5.1.1 : Portion de la vidéo de Big Buck Bunny, *réalisée par la fondation Blender.*

Fig. 5.1.2 : Résultat : Image moyenne de la séquence de Big Buck Bunny *ci-dessus.*

Fig. 5.1.3 : Résultat : Image médiane de la séquence de Big Buck Bunny *ci-dessus.*

En restant dans le domaine du traitement vidéo, remarquons également l'apparition des commandes `-morph_files` et `-morph_video` permettant de calculer des interpolations temporelles de séquences vidéo prenant en compte le mouvement des objets dans les séquences, grâce à un algorithme variationnel et multi-échelle 'estimation de mouvement intra-trames. La **vidéo visible sur YouTube** illustre la différence de rendu pour le recalage d'une séquence, entre une interpolation temporelle utilisant ou non l'estimation du mouvement.

5.2. Déformations et « Glitch Art »

Ceux qui aiment maltraiter leurs images encore plus agressivement seront ravis d'apprendre qu'un tas de nouveaux effets de déformations et de dégradations d'image sont apparus dans G'MIC.

Tout d'abord, le filtre *Deformations / Conformal maps*, qui permet de déformer vos images par des **transformations conformes** [W]. Ces déformations ont la propriété de préserver localement les angles, et sont le plus souvent exprimées par des fonctions utilisant des nombres complexes. En plus de jouer avec des déformations prédéfinies, ce filtre permet aux matheux en herbe d'expérimenter en spécifiant leurs propres formules complexes, si besoin est.

Fig. 5.2.1 : Filtre « Conformal maps » appliquant une transformation conforme à l'image de la Joconde.

Les amateurs de ***Glitch Art*** [W] pourront être également concernés par plusieurs nouveaux traitements dont le rendu fait furieusement penser à des artefacts ou des bogues d'encodage ou de compression d'images. Tout d'abord, l'effet *Degradations / Pixel sort* qui se permet de trier les pixels d'une image par ligne ou par colonne suivant différents critères et sur des régions éventuellement masquées, tel que cela a été proposé et décrit sur ***cette page***.

Fig. 5.2.2 : Filtre « Pixel sort » pour un rendu de type « Glitch Art » du plus bel effet !

Mais aussi avec ses deux petits frères, les filtres *Degradations / Flip & rotate blocs* et *Degradations / Warp by intensity*. Le premier permet de diviser une image en blocs et de tourner ou inverser ces blocs (façon « mirroir ») en ne le faisant potentiellement que pour certaines caractéristiques couleur de l'image (teinte ou saturation par exemple).

Fig. 5.2.3 : Filtre « Flip & rotate blocs » appliqué uniquement sur la teinte afin d'obtenir des artefacts de couleurs façon « Glitch Art ».

Le deuxième s'amuse à déformer localement une image plus ou moins fortement en fonction des informations données par sa géométrie locale et, là aussi, permet de générer des images étranges, mais pouvant être appréciées des amateurs de *Glitch Art*.

Fig .5.2.4 : Filtre « Warp by intensity » appliqué sur l'image de la Joconde (pauvre Joconde !).

Il faut préciser que ces filtres-là ont été très largement inspirés par le greffon ***Polyglitch*** disponible pour ***Paint.NET***, et réalisés après une suggestion d'un sympathique utilisateur (oui, oui, on essaye d'écouter nos utilisateurs les plus sympathiques !).

5.3. Simplification d'image

Qu'avons-nous d'autre en réserve ? Ah oui ! Un nouveau filtre d'abstraction d'image, baptisé *Artistic / Sharp abstract* et basé sur l'algorithme du *Rolling Guidance* déjà évoqué précédemment. Ce filtre applique un lissage à préservation de contours sur une image, et a pour effet principal de supprimer sa texture. La figure ci-dessous illustre son utilisation pour générer plusieurs niveaux d'abstraction d'une même image, à différentes échelles de lissage.

Fig. 5.3.1 : Abstraction d'images via le filtre « Sharp abstract ».

Dans le même genre, G'MIC_ voit également débarquer *Artistic / Posterize*, un effet de **postérisation** [W] d'image. Contrairement au filtre de postérisation livré par défaut dans GIMP, qui essaye principalement de réduire le nombre de couleurs (et qui s'apparente donc plus à un processus de **quantification** [W], notre version ajoute des traitements de simplification des informations spatiales de l'image pour s'approcher un peu plus d'un rendu de type « vieux poster ».

Fig. 5.3.2 : Filtre « Posterize » de G'MIC et comparaison avec le filtre du même nom disponible par défaut dans GIMP.

5.4. Et en vrac...

Si vous n'êtes toujours pas rassasiés (et dans ce cas, on peut dire que vous êtes sacrément gourmands !), nous finissons cette section en extirpant en vrac de nos cartons quelques autres effets nouvellement implémentés.

À commencer par ce filtre *Artistic / Diffusion tensors*, qui affiche un champ de tenseurs de diffusion, calculés à partir des tenseurs de structure d'une image (les tenseurs de structure étant des matrices symétriques et définies positives, classiquement utilisés pour l'estimation de la géométrie locale des structures). Pour être tout à fait honnête, cette fonctionnalité n'avait pas été développée originellement dans un but artistique, mais des utilisateurs du greffon sont tombés dessus un peu par hasard et nous ont demandé d'en faire un filtre pour GIMP, dont acte. Et ma foi, c'est vrai que c'est plutôt joli, non ?

Fig. 5.4.1 : Le filtre « Diffusion tensors » et sa multitude d'ellipses multicolores.

D'un point de vue technique, ce filtre a été en réalité l'occasion d'introduire de nouvelles fonctionnalités de dessin de primitives à l'intérieur de l'évaluateur d'expressions mathématiques de G'MIC, et il est devenu maintenant assez facile de développer ses commandes de visualisation personnalisées pour des données image diverses et variées. C'est ce qui a été fait par exemple avec la commande `-display_quiver`, réimplémentée de zéro, et qui permet de générer ce type de rendu :

Fig. 5.4.2 : Rendu de champs de vecteurs par la commande `-display_quiver`.

Pour les amateurs de textures, nous pouvons mentionner l'arrivée de deux nouveaux effets amusants : D'abord, le filtre *Patterns / Camouflage* qui, comme son nom le suggère, produit une texture de type « camouflage » militaire.

Fig. 5.4.3 : Le filtre « Camouflage », à imprimer sur vos T-shirts pour passer inaperçu en soirée !

Et dans le même genre, le filtre *Patterns / Crystal background* superpose plusieurs polygones colorés aléatoires pour synthétiser une texture faisant vaguement penser à du cristal vu à la loupe. Plutôt sympa quand on veut obtenir rapidement des fonds d'images pas trop ternes.

Fig. 5.4.4 : Le filtre « Crystal background » en action.

Et nous finirons ce tour d'horizon des nouveaux traitements apparus depuis l'an dernier avec l'effet *Rendering / Barnsley fern*, qui trace un rendu de la fractale **Barnsley fern** [W]. Pour les curieux, notons que l'algorithme de génération de cette figure fractale notoire est disponible sur ***Rosetta Code***, avec même une version du code écrite en langage de script G'MIC, à savoir :

```
# Put this into a new file 'fern.gmic' and invoke it from the command line, like this:
# $ gmic fern.gmic -barnsley_fern
```

```
barnsley_fern :
  1024,2048
  -skip {"
      f1 = [ 0,0,0,0.16 ];           g1 = [ 0,0 ];
      f2 = [ 0.2,-0.26,0.23,0.22 ];  g2 = [ 0,1.6 ];
      f3 = [ -0.15,0.28,0.26,0.24 ]; g3 = [ 0,0.44 ];
      f4 = [ 0.85,0.04,-0.04,0.85 ]; g4 = [ 0,1.6 ];
      xy = [ 0,0 ];
      for (n = 0, n<2e6, ++n,
        r = u(100);
        xy = r<=1?((f1**xy)+=g1):
             r<=8?((f2**xy)+=g2):
             r<=15?((f3**xy)+=g3):
                   ((f4**xy)+=g4);
        uv = xy*200 + [ 480,0 ];
        uv[1] = h - uv[1];
        I(uv) = 0.7*I(uv) + 0.3*255;
      )"}
  -r 40%,40%,1,1,2
```

Et voici le rendu que cette fonction génère :

Fig. 5.4.5 : Rendu de la fractale « Barnsley fern » calculé par G'MIC.

6. Autres informations générales

Bien évidemment, les filtres qui ont été présentés tout au long de cette dépêche ne constituent que la partie émergée de l'iceberg G'MIC. Ils sont en réalité le résultat de nombreux développements et améliorations effectués « sous le capot », c.-à-d. directement sur le code de l'interpréteur du **langage de script** que G'MIC définit, et qui sert *in fine* de base à l'écriture de tous les filtres et commandes disponibles pour les utilisateurs. Durant cette année écoulée, un gros travail a donc été réalisé pour améliorer les performances et les possibilités de cet interpréteur :

- l'évaluateur d'expressions mathématiques a été considérablement enrichi et optimisé, avec toujours plus de fonctions disponibles (particulièrement pour le calcul matriciel), la gestion des chaînes de caractères, l'introduction de variables `const` pour une plus grande rapidité d'évaluation, la possibilité d'écrire des macros **variadiques**, ou encore d'allouer des vecteurs de manière dynamique, etc. ;
- de nouvelles optimisations ont également été introduites dans la bibliothèque **CImg**, avec notamment la parallélisation de nouvelles fonctions (via l'utilisation d'**OpenMP**[W]). Cette bibliothèque *C++* fournit les implémentations des traitements d'images « critiques » et son optimisation a donc des répercussions directes sur les performances de G'MIC (à ce propos, notons que CImg passe également en version majeure 2.0) ;
- la compilation de G'MIC sous Windows utilise maintenant une version de `g++` récente (la 6.2 plutôt que la 4.5), ce qui se traduit par un impact énorme sur les performances des exécutables compilés ; certains filtres s'exécutent jusqu'à 60 fois plus rapidement qu'avec les binaires générés précédemment (c'est le cas, par exemple, pour le filtre *Deformations / Conformal Maps*, évoqué en section 5.2) ;
- la prise en compte des images `.tiff` de grande dimension (format **BigTIFF**, avec des fichiers dont la taille peut dépasser les 4 Gio) est maintenant opérationnelle en lecture et écriture, tout comme celle des images TIFF à valeurs flottantes sur 64 bits ;
- le moteur de rendu d'objets 3D maillés intégré à G'MIC a également été amélioré, avec la prise en charge du ***bump mapping***[W] ; aucun filtre n'utilise actuellement cette fonctionnalité, mais sait-on jamais, on se prépare pour la suite !

Fig. 6.1 : Comparaison du rendu texturé avec (à droite) et sans « Bump mapping » (à gauche).

- et comme il est toujours bon de se détendre après une dure journée de travail, nous avons ajouté le jeu du **Puissance 4**[W] à G'MIC ! Il est accessible via la commande shell `gmic -x_connect4` ou en passant par le greffon, via le filtre *Various / Games & demos / Connect-4*. À noter qu'il est possible de jouer contre l'ordinateur, qui a un niveau décent mais pas imbattable non plus (l'IA, très simple, utilise l'**algorithme Minimax**[W] avec un arbre de décisions à deux niveaux seulement).
  Fig. 6.2 : Le jeu du « Puissance 4 », jouable dans G'MIC.

Et pour en finir avec le rayon des nouveautés, mentionnons le travail en cours de refonte du service Web *G'MIC Online*, avec une **version bêta** en cours de développement. Ce re-développement du site, réalisé par **Christophe Couronne**, chapeauté par **Véronique Robert** (tous deux également membres du laboratoire *GREYC*), a été étudié pour mieux s'adapter aux appareils mobiles, et les premiers tests sont plus qu'encourageants. N'hésitez pas à expérimenter et nous faire part de vos impressions !

7. La suite ?

Que retenir de cette looooooongue dépêche ?

D'abord que cette version 2.0 représente effectivement une étape importante dans la vie du projet G'MIC, et que les améliorations réalisées ces derniers mois sont de bon augure pour les développements futurs. Il semble que les utilisateurs soient de plus en plus nombreux (et apparemment satisfaits), et nous espérons que cela incitera d'autres développeurs à déployer notre nouvelle interface G'MIC-Qt comme greffon pour leurs propres logiciels libres. En particulier, on a bon espoir de voir tourner le nouveau G'MIC sous Krita prochainement, et ce serait déjà une étape formidable !

Ensuite, que ce projet libre continue d'être actif et d'évoluer au fil des rencontres et des discussions avec les membres des communautés d'artistes et de photographes, qui peuplent en particulier les forums et IRC de ***Pixls.us*** et de **GimpChat**. Vous avez de grandes chances de nous y retrouver si vous désirez de plus amples informations ou simplement discuter de choses relatives au traitement d'images (libre de préférence).

On tient, bien sûr, à remercier les contributeurs du projet, que l'on ne peut malheureusement pas tous citer ici, mais qui sont de plus en plus nombreux. Merci également à tous ceux qui payent leurs impôts en France sans ronchonner et qui font donc avancer indirectement le développement de ce projet libre, mis à disposition gracieusement — faut-il le rappeler ? — par le **GREYC**, un laboratoire public de recherche en **STIC** — Sciences et technologies d'information et de communication — de Caen, et qui est développé par des fonctionnaires motivés et passionnés (et du coup quelquefois aussi fatigués ! :)).

Et en attendant une prochaine dépêche hypothétique sur une future version de G'MIC, vous pouvez toujours suivre l'avancement pas à pas du projet via **notre fil Twitter**.

D'ici là, vive le traitement d'images libre et la science reproductible !

Crédits images : Sauf mention contraire, les diverses images non synthétiques qui ont servi à illustrer cet article proviennent du site **Pixabay**.

Aller plus loin

- **Page principale du projet G'MIC** (1233 clics)
- **Fil Twitter** (112 clics)
- **Le greffon pour GIMP** (751 clics)
- **Le Web service G'MIC Online** (196 clics)
- **Le journal des modifications de la version 2.0.0** (194 clics)
- **Série d'articles G'MIC sur LinuxFr.org** (464 clics)