G'MIC 1.7.1 : quand les fleurs bourgeonnent, les filtres d'images foisonnent.

Posté par **David Tschumperlé** (**site web personnel**) le 05/05/16 à 10:57. Édité par **5 contributeurs**. Modéré par **Xavier Claude**.
Licence CC By-SA.
Étiquettes : **gimp** , **krita** , **g'mic** , **traitement_d'images** **+ Étiqueter**

La version **1.7.1** « *Spring 2016* » de ***G'MIC*** (*GREYC's Magic for Image Computing*), infrastructure libre pour le traitement d'images, a été publiée récemment, le 26 avril 2016. Nous continuons notre **série de présentation** des possibilités et des avancées de ce logiciel libre, avec la description des nouveautés et des améliorations notables introduites depuis **notre dernière dépêche** sur ce sujet, datant de décembre 2015, qui avait été rédigée à l'occasion de la sortie de la version *1.6.8*. Trois versions successives ont été publiées depuis (les versions *1.6.9*, *1.7.0* et *1.7.1*).
La deuxième partie de la dépêche détaille quelques uns des nouveaux filtres et effets disponibles dans le **greffon *G'MIC*** pour ***GIMP***, qui reste l'interface de *G'MIC* la plus utilisée à ce jour. Nous abordons aussi les autres évolutions diverses du projet comme l'amélioration et la création d'autres interfaces d'utilisation ainsi que les avancées « techniques » réalisées au cœur du *framework*.

Sommaire

- **1. Le projet *G'MIC* en quelques mots**
- **2. Sélection de nouveaux filtres et effets**
  - **2.1. Création de peintures à partir de photographies**
  - **2.2. Reconstruction de données manquantes à partir d'échantillons épars**
  - **2.3. Rendre des textures périodiques**
  - **2.4. Décomposition d'une image en niveaux de détails**
  - **2.5. Débruitage d'images par méthode « Patch-PCA »**
  - **2.6. Effet « Droste » : la mise en abyme continue**
  - **2.7. Transformation équirectangulaire <-> zénith/nadir**
- **3. Autres améliorations et faits notables**
- **4. Comment tout cela va évoluer ?**

1. Le projet *G'MIC* en quelques mots

G'MIC est un projet libre ayant vu le jour en août 2008, dans l'équipe ***IMAGE*** du laboratoire ***GREYC*** (Unité Mixte de Recherche du ***CNRS*** située à Caen / France). Cette équipe est composée de chercheurs et d'enseignant-chercheurs spécialisés dans les domaines de l'algorithmique et des mathématiques du traitement d'images. *G'MIC* est distribué sous licence libre ***CeCILL*** (compatible *GPL*) pour différentes plateformes (*Linux, Mac et Windows*). Il fournit un ensemble d'interfaces utilisateurs variées pour la manipulation de données images *génériques*, à savoir des images ou des séquences d'**images hyperspectrales** [W] 2D ou 3D à valeurs flottantes (ce qui inclut bien évidemment les images couleurs « classiques »).

Fig.1.1. Logo et (nouvelle) mascotte du projet G'MIC, logiciel libre pour le traitement d'image.

Notons qu'une première nouveauté relative au projet concerne *Gmicky*, la mascotte, qui a été entièrement redessinée, par ***David Revoy***, artiste français bien connu des amoureux du graphisme libre, puisqu'il est à l'origine du fameux webcomics ***Pepper&Carott***. Un grand merci à lui ! (à comparer avec l'ancien dessin de *Gmicky* toujours visible **ici**).

G'MIC s'est fait connaître essentiellement via son **greffon** disponible pour le logiciel ***GIMP***, apparu en 2009, greffon qui propose plus de *460* différents filtres et effets à appliquer sur vos images, et qui ressemble aujourd'hui à ceci :

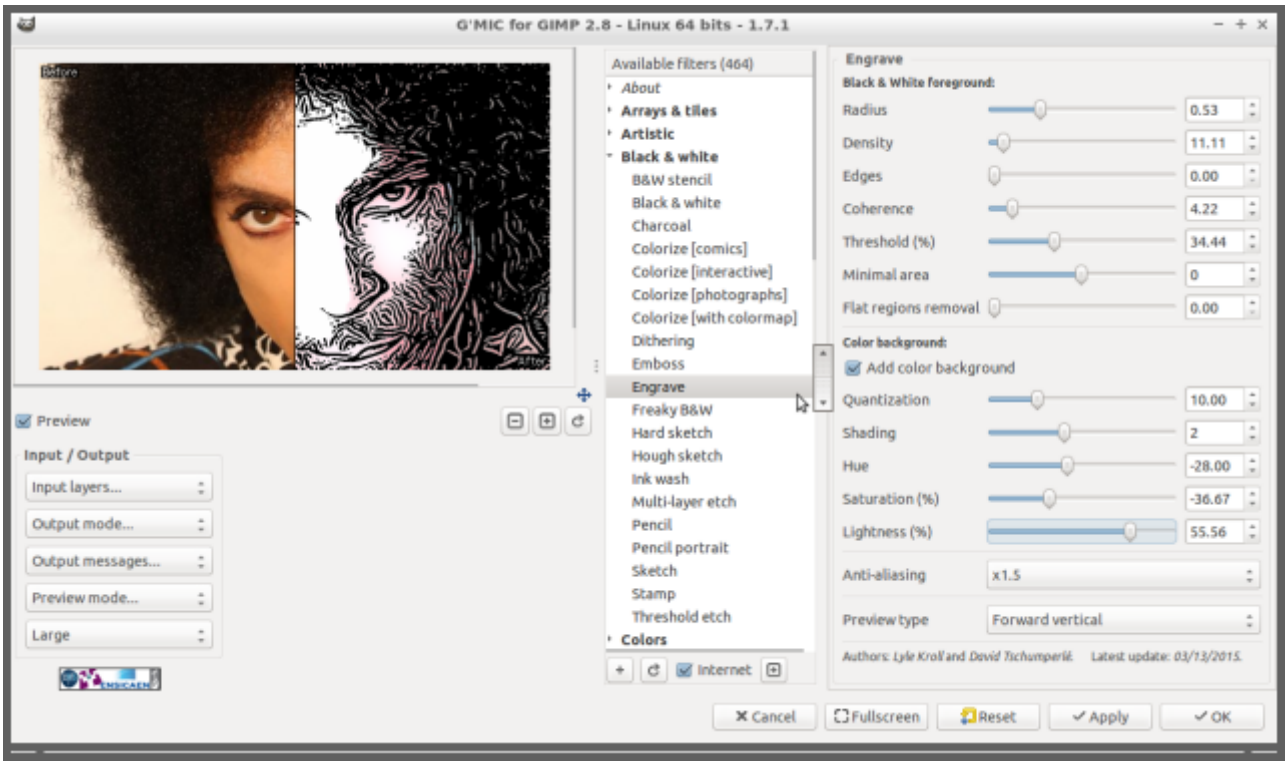

Fig.1.2. Aperçu de la version 1.7.1 du greffon G'MIC pour GIMP.

Mais *G'MIC* n'est pas qu'un greffon pour GIMP. Il fournit également une interface en **ligne de commande**, qui peut s'utiliser de manière complémentaire aux outils *CLI* proposés par ***ImageMagick*** ou ***GraphicsMagick***. Cette interface *CLI* est, comme on peut l'imaginer, l'interface la plus puissante et la plus flexible du *framework*. Il existe aussi un service web ***G'MIC Online*** associé, pour appliquer des effets sur vos images directement à partir d'un navigateur web. D'autres interfaces basées sur *G'MIC* sont également développées (***ZArt***, un greffon pour ***Krita***, des filtres pour ***Photoflow***…) mais leur usage reste pour le moment plus confidentiel. Toutes ces interfaces se basent sur les bibliothèques *C++* ***CImg*** et ***libgmic*** qui sont portables, thread-safe et multi-threadées (via l'utilisation d'***OpenMP***). Aujourd'hui, *G'MIC* possède plus de ***900 fonctions*** différentes de traitement d'images, toutes paramétrables, pour une bibliothèque de seulement *6 Mio* correspondant à un peu plus de *150 kloc* 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 / le recalage, l'affichage de primitives (jusqu'aux objets 3d maillés), la détection de contours/la segmentation, le rendu artistique, etc. C'est donc un outil très générique aux usages variés, très utile d'une part pour convertir, visualiser et explorer des données images, et d'autre part pour construire des pipelines personnalisés et élaborés de traitements d'images (voir **ces transparents de présentation du projet** pour plus d'information sur les motivations et les buts de ce projet).

2. Sélection de nouveaux filtres et effets

Nous proposons ici un résumé des nouveaux filtres et effets les plus marquants récemment développés, et illustrons leur usage depuis le greffon *G'MIC* pour *GIMP*. Ces filtres sont bien sûr utilisables également depuis les autres interfaces disponibles (notamment avec `gmic`, l'interface en ligne de commande). Nous nous sommes restreints aux filtres les plus intéressants à expliquer et illustrer, car en réalité, ce sont plus d'une vingtaine de nouveaux filtres et effets qui ont fait leur apparition depuis la version *1.6.8*.

2.1. Création de peintures à partir de photographies

Le filtre **Artistic / Brushify** tente de transformer une image en *peinture*. L'idée ici est de simuler (de manière simplifiée) le processus de création d'une peinture sur une toile blanche. On fournit une image *modèle* à l'algorithme, qui va dans un premier temps, analyser sa géométrie (principalement le contraste et l'orientation des contours), puis tenter de la *repeindre* en utilisant comme outil un unique pinceau (*brush* en anglais, d'où le nom de l'effet) qui va s'orienter localement pour s'adapter à la géométrie des contours de l'image.
En simulant suffisamment de coups de pinceaux, on obtient une image « peinte » plus ou moins fidèle à l'image modèle d'origine, en fonction de la forme et de la taille du pinceau utilisé, du nombre d'orientations autorisées, etc. Tout ceci étant réglable par l'utilisateur comme des paramètres de l'algorithme : ce filtre permet donc d'obtenir une grande variété de rendus différents.

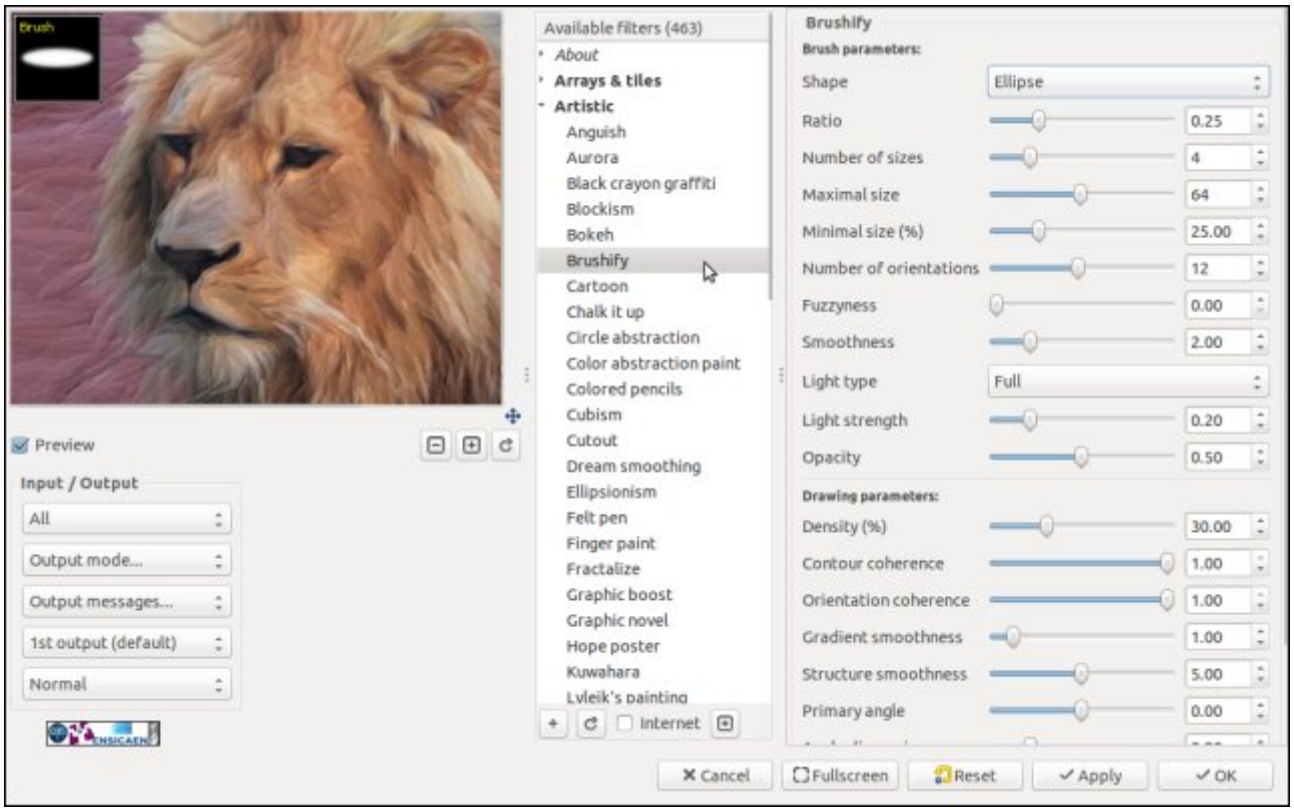

Fig.2.1.1. Apercu du filtre « Brushify » dans le greffon G'MIC pour GIMP. La brosse qui va être utilisée par l'algorithme est visible en haut à gauche.

L'animation ci-dessous illustre cette grande diversité de résultats, avec le traitement d'une même image d'entrée (photographie d'une tête de lion), en variant les types de brosses et les paramètres utilisés par l'algorithme. *Brushify* peut être assez coûteux en termes de temps de calcul (suivant le nombre de coups de pinceaux à simuler), même si l'implémentation de l'algorithme est déjà parallélisée (les différents coeurs de calcul pouvant donner des coups de pinceaux simultanément).

Fig.2.1.2. Quelques exemples de rendus du filtre « Brushify » à partir d'une même image d'entrée, en utilisant des brosses différentes.

À noter qu'il est amusant d'invoquer ce filtre à partir de la ligne de commande (grâce à la fonction `-brushify` disponible dans `gmic`), pour traiter des lots d'images et des vidéos (**un exemple de vidéo « brushifiée »**).

2.2. Reconstruction de données manquantes à partir d'échantillons épars

G'MIC se dote d'une nouvelle fonctionnalité de reconstruction de données manquantes dans des images. Nous avons déjà évoqué ce problème classique de reconstruction en traitement d'images dans des dépêches précédentes (avec l'*inpainting* comme illustré **ici** ou encore **ici**). La nouvelle méthode d'interpolation ajoutée suppose quant à elle que l'on ne dispose que de *données connues éparses*, par exemple quelques pixels de couleurs dispersés ça et là dans l'image, plutôt que des blocs entiers de données contiguës connues. L'analyse et la reconstruction de la géométrie des structures présentes dans l'image devient alors un problème particulièrement difficile.

La nouvelle fonction `-solidify` de *G'MIC* permet de reconstruire des données images denses à partir de quelques points épars connus, en utilisant une technique de reconstruction multi-échelle basée sur les ***EDP***[W] de diffusion. La figure ci-dessous illustre les capacités de cette méthode, avec un exemple de reconstruction d'une **image de goutte d'eau**. On ne garde ici que 2,7% des données image (ce qui est vraiment peu !) et l'algorithme reconstruit une image entière, qui ressemble à celle d'origine (même si, bien entendu, tous les détails de l'image originale n'ont pas été reconstruits complètement). Plus on dispose d'échantillons, plus on est capable de reconstruire des détails.

Fig.2.2.1. Reconstruction d'une image à partir d'un échantillonnage épars.

Cette technique de reconstruction étant assez générique, plusieurs filtres différents se basant sur celle-ci ont pu être élaborés et ajoutés dans *G'MIC* :

- Le filtre **Repair / Solidify** permet d'appliquer l'algorithme de reconstruction de manière directe, en reconstruisant par interpolation les zones marquées comme transparentes dans les images d'entrées. L'animation ci-dessous montre l'application de ce filtre pour la réalisation d'un effet de flou artistique sur les bords d'une image.

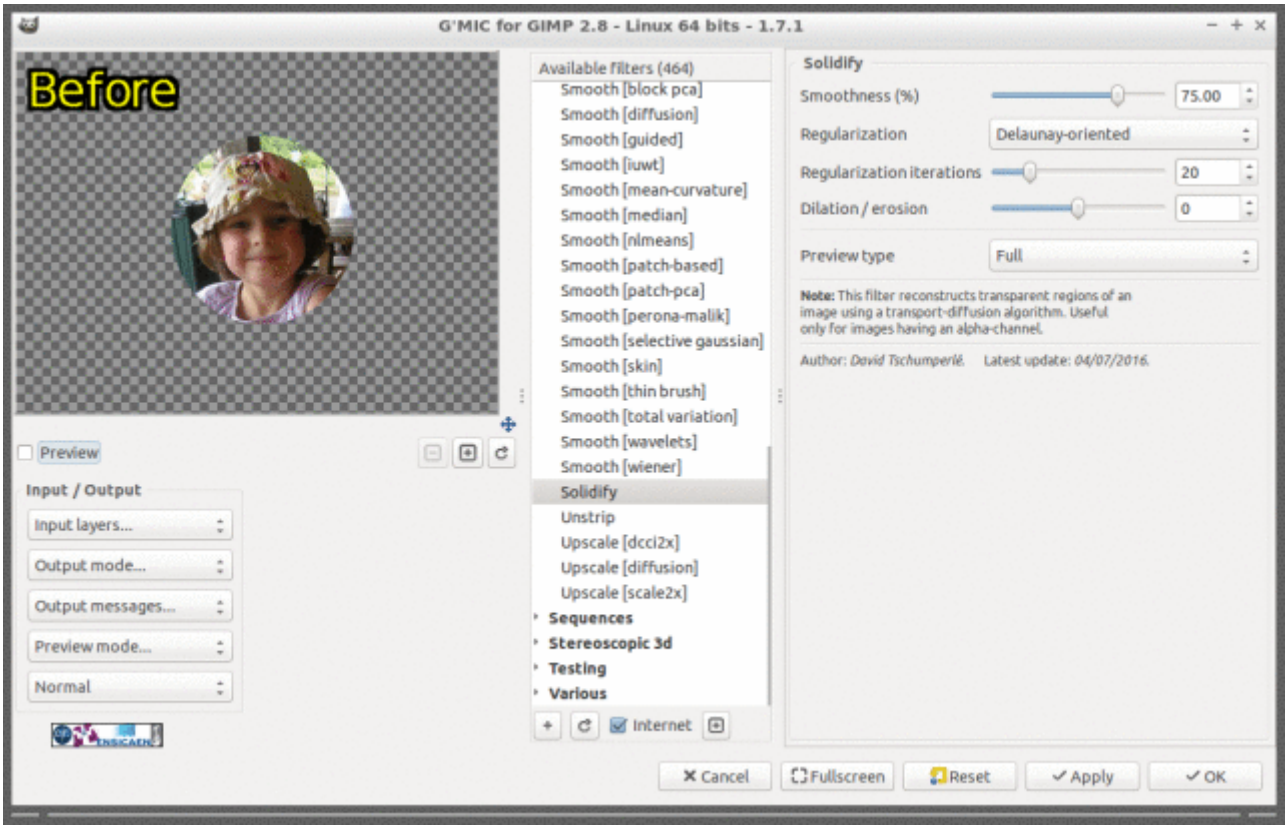

Fig.2.2.2. Aperçu du filtre « Solidify » dans le greffon G'MIC pour GIMP.

D'un point de vue artistique, les possibilités de ce filtre sont nombreuses. Il est par exemple très utile pour générer simplement des dégradés de couleurs de formes complexes dans des images, comme le montre les deux

exemples de la figure ci-dessous (ou encore **cette vidéo**, qui détaille le processus)

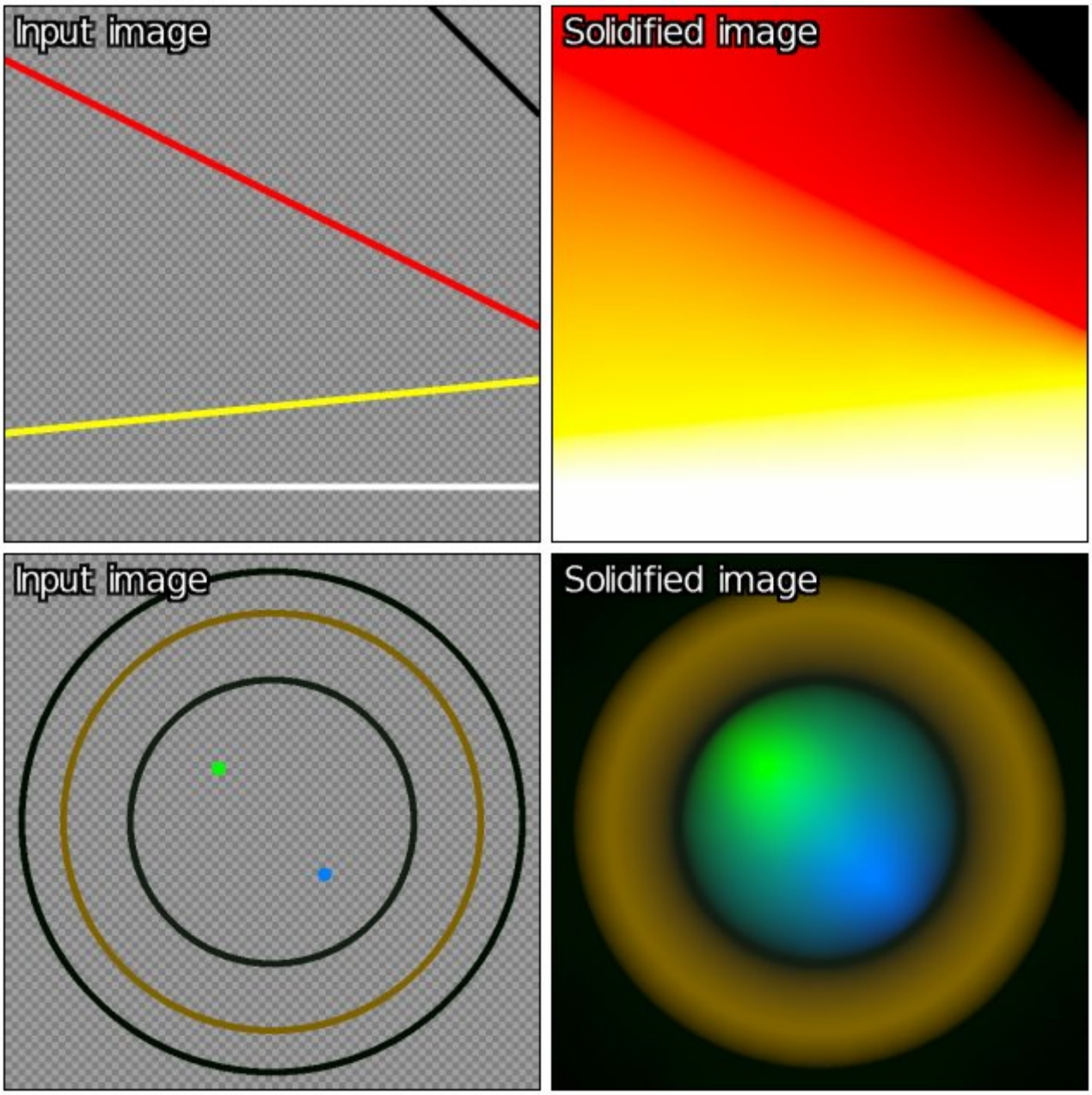

Fig.2.2.3. Utilisation du filtre « Solidify » de G'MIC pour créer simplement des dégradés de couleurs aux structures géométriques complexes (images d'entrées à gauche, résultats du filtre à droite).

- Le filtre **Artistic / Smooth abstract** reprend le principe utilisé pour l'image de la goutte d'eau vu précédemment : il échantillonne une image de manière éparse, en plaçant des points clés préférentiellement sur les contours présents dans celle-ci, puis tente de reconstruire l'image entière à partir de ces échantillons seuls. Si le nombre d'échantillons est faible, le filtre va générer une image *continue par morceaux* qui peut donc être vue comme une abstraction lisse de l'image d'origine (voir la figure ci-dessous).

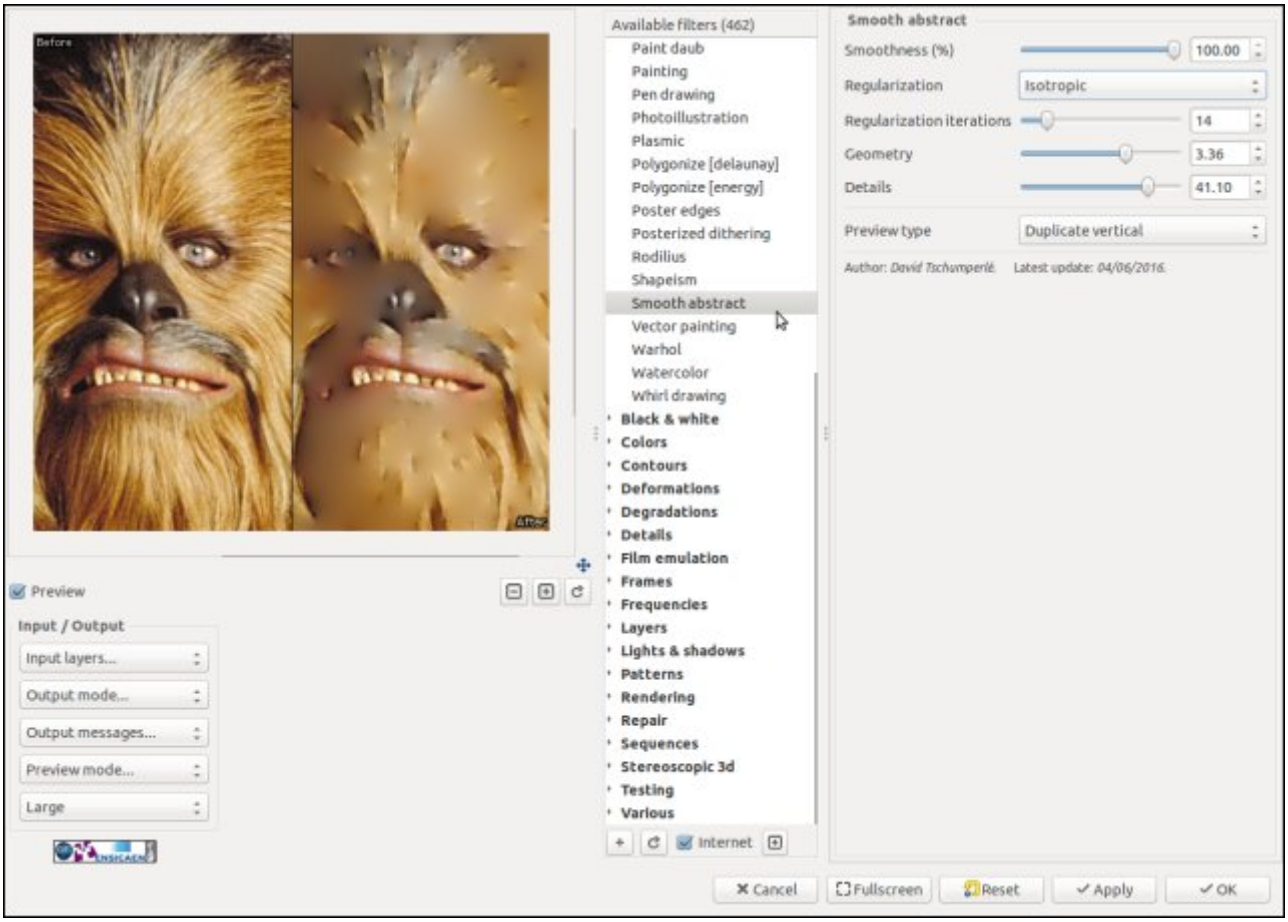

Fig.2.2.4. Aperçu du filtre « Smooth abstract » dans le greffon G'MIC pour GIMP.

- Le filtre **Rendering / Gradient [random]** permet quant à lui la création de fonds colorés. Le filtre plaçe des points de couleurs aléatoirement sur une image, et les interpole ensuite spatialement avec l'algorithme de reconstruction. On obtient facilement des fonds d'écrans psychédéliques composés de dégradés de couleurs qui partent dans toutes les directions (voir figure ci-dessous).

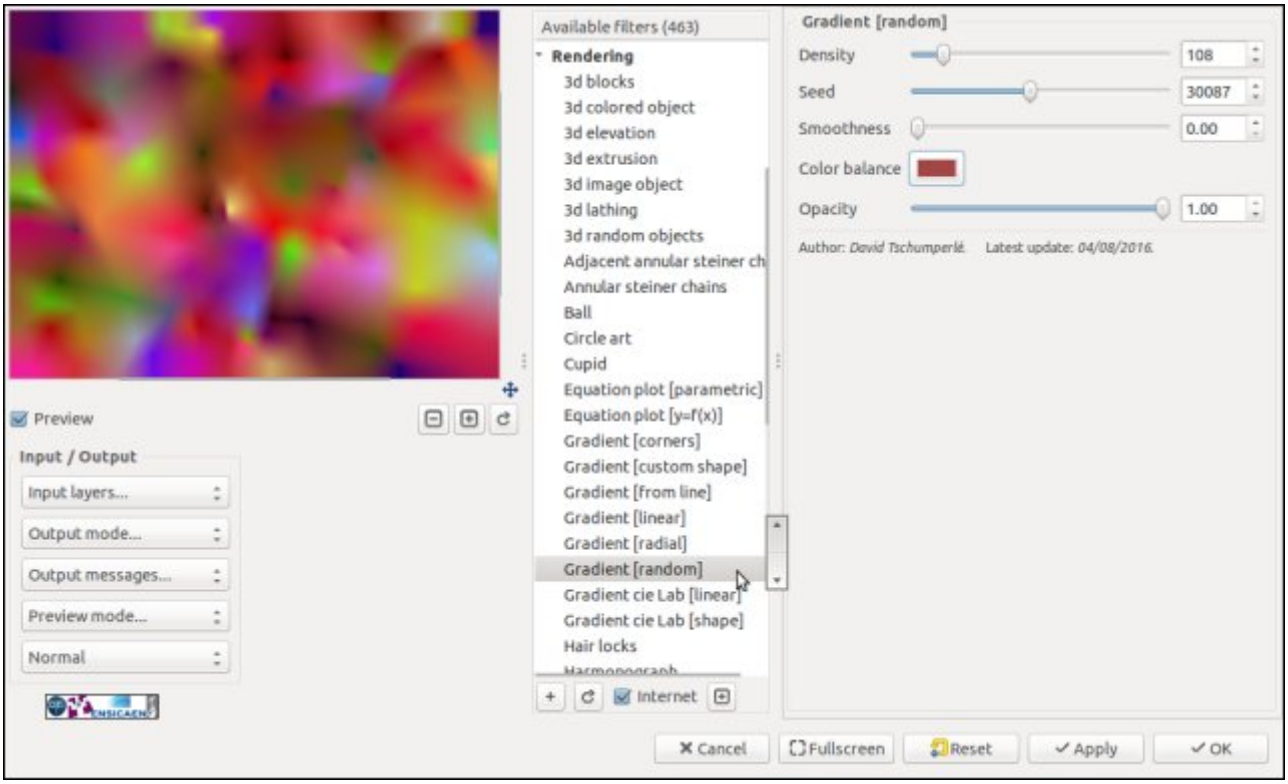

Fig.2.2.5. Aperçu du filtre « Gradient [random] » dans le greffon G'MIC pour GIMP.

- **Simulation de films argentiques** : ce nouvel algorithme de reconstruction d'images à partir d'échantillons épars a également une grande utilité pour les nombreux filtres de simulation de films argentiques, présents dans *G'MIC* depuis quelques années déjà. La section **Film emulation** propose en effet un grand choix de filtres dont le but est d'appliquer des transformations colorimétriques, pour simuler le rendu qu'aurait eu une photo numérique si elle avait été prise avec un appareil argentique muni d'un certain type de pellicule. La figure ci-dessous montre par exemple quelques unes des *300* transformations colorimétriques qu'il est possible d'appliquer à partir de *G'MIC*.

Fig.2.2.6. Quelques unes des transformations colorimétriques disponibles dans G'MIC (parmi + de 300).

D'un point de vue algorithmique, ces algorithmes de transformation colorimétrique sont très simples à mettre en œuvre : on dispose pour chacune des 300 transformations d'une ***HaldCLUT***, c'est-à-dire d'une fonction définissant pour chaque couleur *(R,G,B)* des pixels de l'image originale, une nouvelle couleur *(R,G,B)* à attribuer aux pixels de l'image résultante. Cette fonction n'étant pas forcément analytique, une *HaldCLUT* est généralement stockée sous forme discrétisée, et donne donc le résultat de la transformation colorimétrique *pour toutes les couleurs possibles* du cube *RGB* (soit $2^{24} = 16777216$ valeurs si on travaille avec une précision de *8bits* par composante). La

figure ci-dessous illustre la façon dont une transformation colorimétrique basée *HaldCLUT* s'applique sur l'ensemble des couleurs du cube *RGB*.

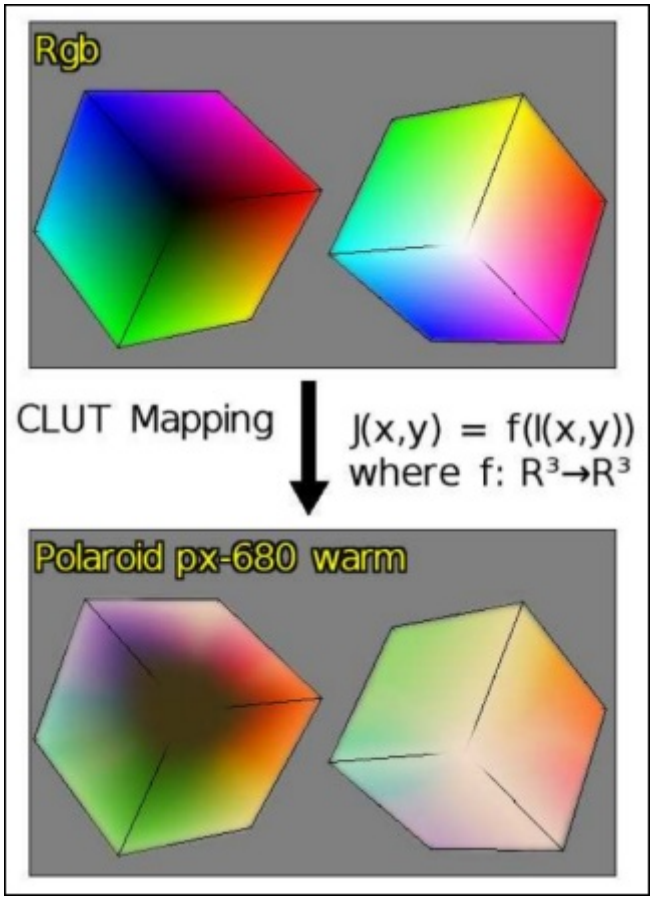

Fig.2.2.7. Principe d'une transformation colorimétrique utilisant une HaldCLUT.

Autant dire que, même en sous-échantillonnant l'espace *RGB* (sur *6 bits* par composante par exemple) et en compressant sans perte le fichier de transformation colorimétrique correspondant, on se retrouve vite avec un fichier qui est relativement volumineux (entre *200* et *300* Kio par fichier). Multipliez ce nombre par *300* (le nombre de transformations colorimétriques disponibles dans *G'MIC*), et vous arrivez à un total de *85 Mio* environ pour stocker l'ensemble de ces transformations. Un peu lourd à diffuser pour de simples filtres de changement de couleurs !

L'idée était donc de développer une méthode de compression avec pertes qui pourrait s'adapter spécifiquement aux données de type *HaldCLUT*, c'est-à-dire à des fonctions volumiques discrétisées à valeurs vectorielles, qui sont par nature relativement lisses par morceaux. C'est donc ce qui a été fait, en se basant sur l'algorithme de reconstruction de données images à partir d'échantillons épars. Cet algorithme fonctionne en effet avec des données images pouvant être volumiques. Il suffit donc d'extraire un nombre suffisant de points-clés significatifs dans le cube *RGB* pour permettre la reconstruction d'une *HaldCLUT* entière, avec une erreur de reconstruction suffisamment faible pour que le résultat de la transformation colorimétrique résultante soit indistinguable de la transformation colorimétrique originale.

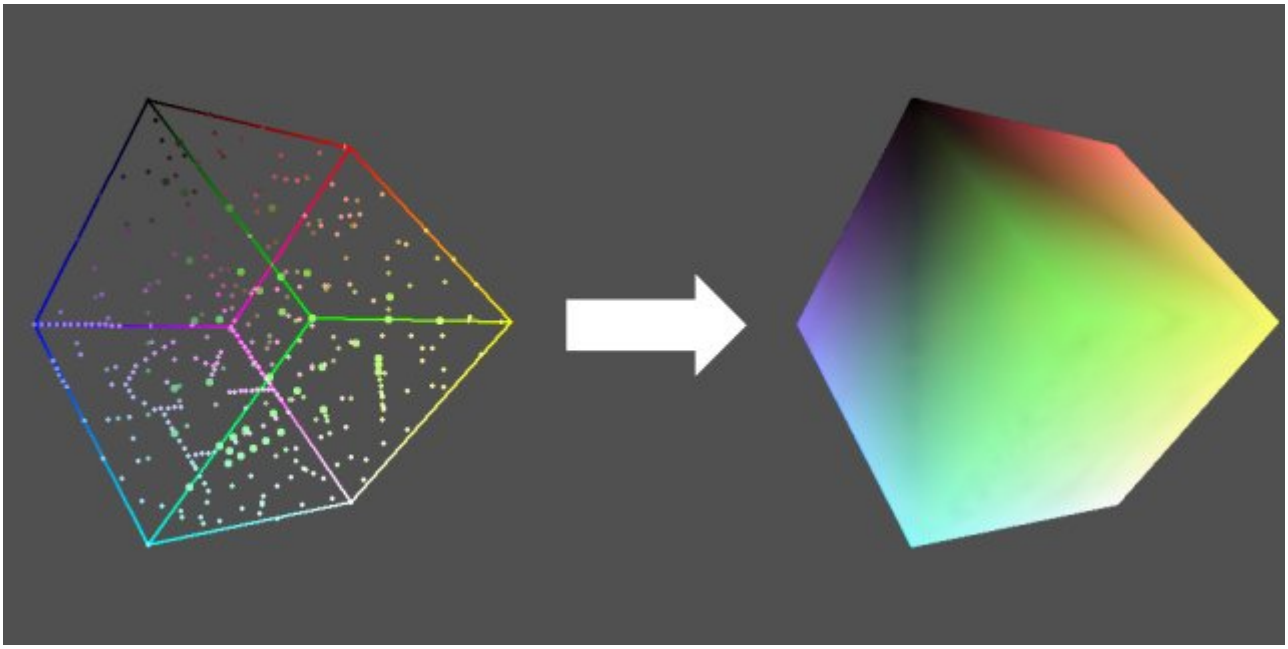

Fig.2.2.8. Principe du codage et de la reconstruction d'une HaldCLUT à partir d'un nuage de points clés définit dans le cube RGB.

Donc, au lieu de stocker l'ensemble des couleurs d'une *HaldCLUT*, on n'en stocke plus qu'un sous-ensemble épars représenté par une liste de *{ point-clés, couleurs }*, et on laisse l'algorithme de reconstruction faire son travail pour regénérer la *HaldCLUT* entière, avant de l'appliquer sur l'image à modifier. Suivant la complexité des *HaldCLUTs* à appliquer, plus ou moins de points clés sont nécessaires (ça peut varier de *30* à *2000*).
Résultat des courses : On passe de *85 Mio* pour le stockage des *300 HaldCLUTs* de *G'MIC* à *850 Kio*, soit un gain de compression de *99%* ! D'un point de vue pratique, ce nouveau fichier décrivant toutes les *HaldCLUTs* compressées est facilement distribuable et installable avec le greffon, et un utilisateur peut donc appliquer toutes les transformations colorimétriques de *G'MIC* en restant *hors-ligne* (alors qu'auparavant, chaque *HaldCIUT* était téléchargée lors de l'application d'une nouvelle transformation colorimétrique).

Ce nouvel algorithme de reconstruction d'images à partir d'échantillons épars a donc beaucoup d'intérêt, et nul doute qu'il sera réutilisé dans d'autres filtres prochainement.

2.3. Rendre des textures périodiques

Le filtre **Arrays & tiles / Make seamless [patch-based]** permet de transformer une texture d'entrée en la rendant *tuilable*, c'est-à-dire en permettant sa répétition sous forme de *tuiles* le long des axes horizontaux et verticaux, sans que l'on distingue de discontinuités visibles lorsque les bords de deux tuiles adjacentes sont mises bout à bout. C'est une opération qui peut s'avérer très difficile à réaliser si la texture d'entrée est complexe, par exemple avec peu d'auto-similarité, ou avec des changements de luminosités flagrants.
C'est le cas de l'exemple illustré dans la figure ci-dessous, avec une texture *chair de saumon* présentée sous forme de 4 tuiles disposées en configuration *2x2*. L'éclairage de cette texture varie de gauche à droite (du plus sombre vers le plus clair). L'algorithme proposé permet ici de transformer la texture pour que le recollement devienne quasiment invisible. Notons que l'on cherche ici à préserver la texture d'entrée le plus possible, contrairement à l'algorithme de **re-synthèse de texture** qui était déja disponible, et qui cherchait plutôt à recréer de toute pièces une instance aléatoire d'une texture de taille quelconque ayant les mêmes caractéristiques que la texture modèle. Essayez de réaliser ceci manuellement, et vous vous rendrez compte de la difficulté du problème (qui pourrait paraître simple au premier abord).

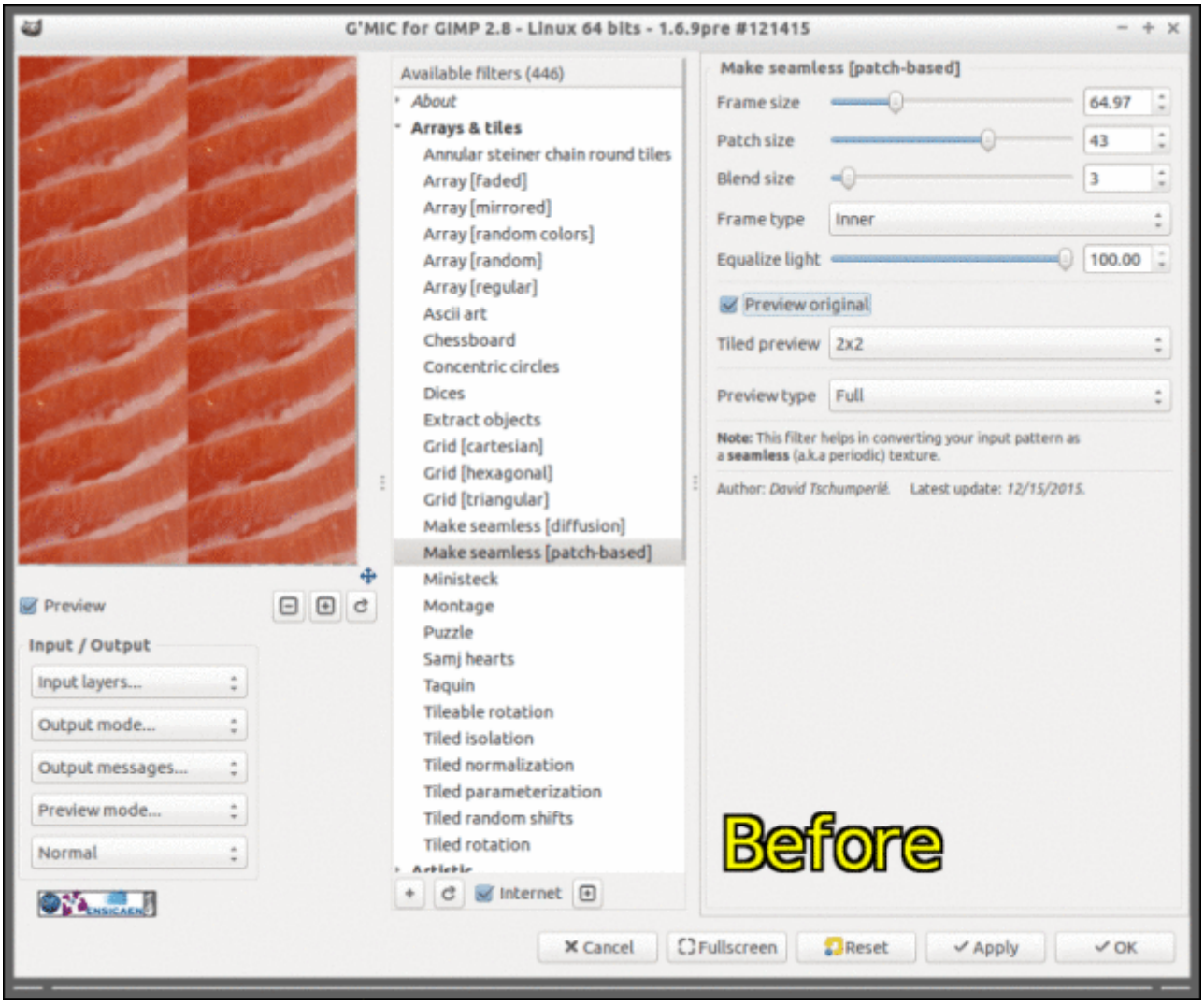

Fig.2.3.1. Aperçu du filtre « Make Seamless » du greffon G'MIC pour GIMP, pour rendre des textures tuilables.

À noter que la création de ce nouveau filtre d'aide au tuilage a été suggérée par **rewind** dans **les commentaires** de la dépêche précédente sur G'MIC ! Les grands esprits se rencontrent sur *LinuxFr.org* :)
On peut imaginer de belles applications à ce type de filtres, notamment dans le domaine du jeu vidéo où tuiler des textures pour créer de grands mondes virtuels est monnaie courante. Un autre exemple de tuilage d'une texture complexe de mousse en (tuilage *2x2*) est présenté dans l'animation ci-dessous.

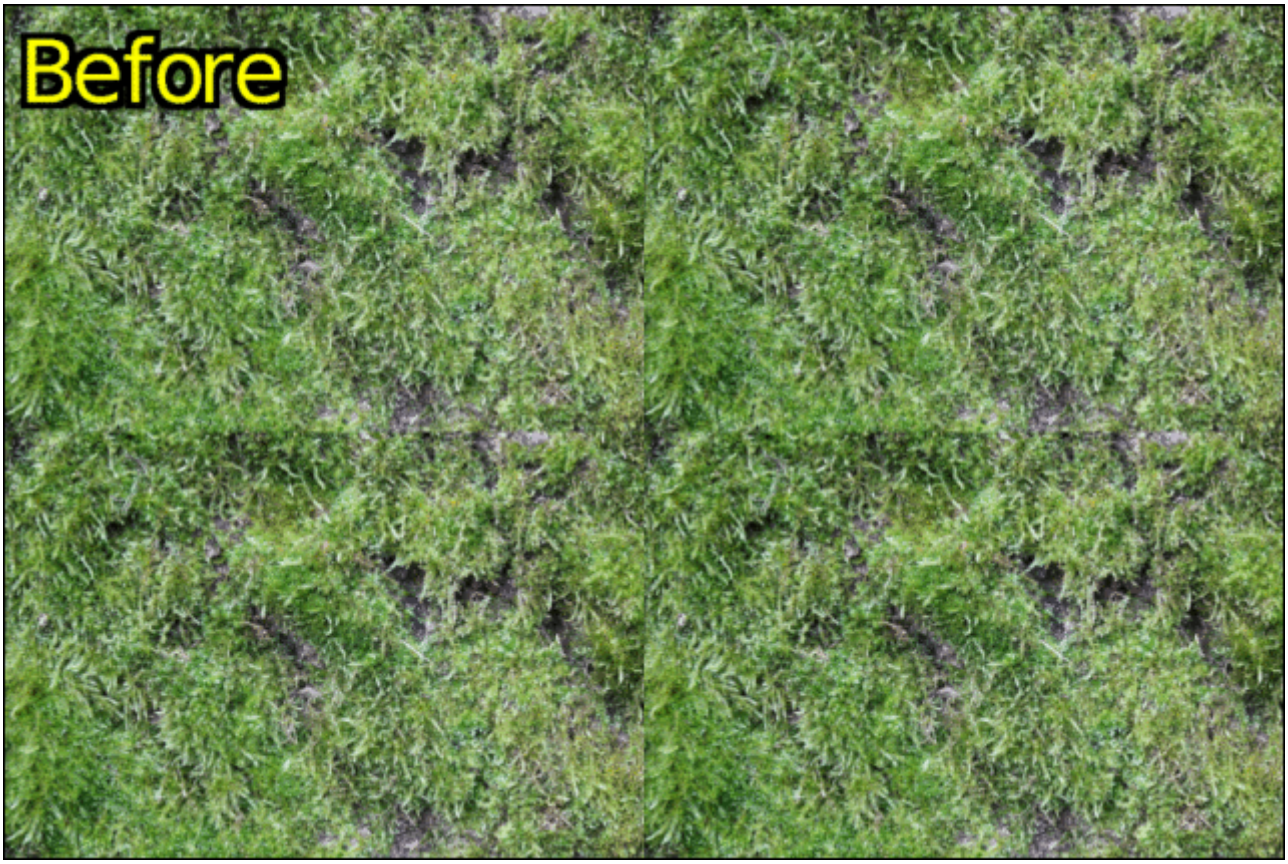

Fig.2.3.2. Résultat du filtre « Make seamless » de G'MIC pour rendre tuilable une texture de mousse.

2.4. Décomposition d'une image en niveaux de détails

Un nouveau filtre de décomposition d'image en plusieurs niveaux de détails nommé **Details / Split details [wavelets]** a également été ajouté. Il implémente un algorithme de décomposition en ondelettes à trous. Pour les connaisseurs, c'est exactement le même algorithme que celui qui est proposé dans le greffon populaire ***Wavelet Decompose*** pour *GIMP*, avec ici en plus, une prévisualisation des échelles de détails et une implémentation parallélisée, tirant parti du multi-coeurs. La figure ci-dessous illustre son action sur un portrait. L'application de ce filtre décompose une image en plusieurs calques de sortie, de telle manière à ce que chaque calque contienne les détails de l'image à une échelle donnée, et que l'ensemble de ces calques de sortie superposés redonne bien évidemment le rendu de l'image d'origine.

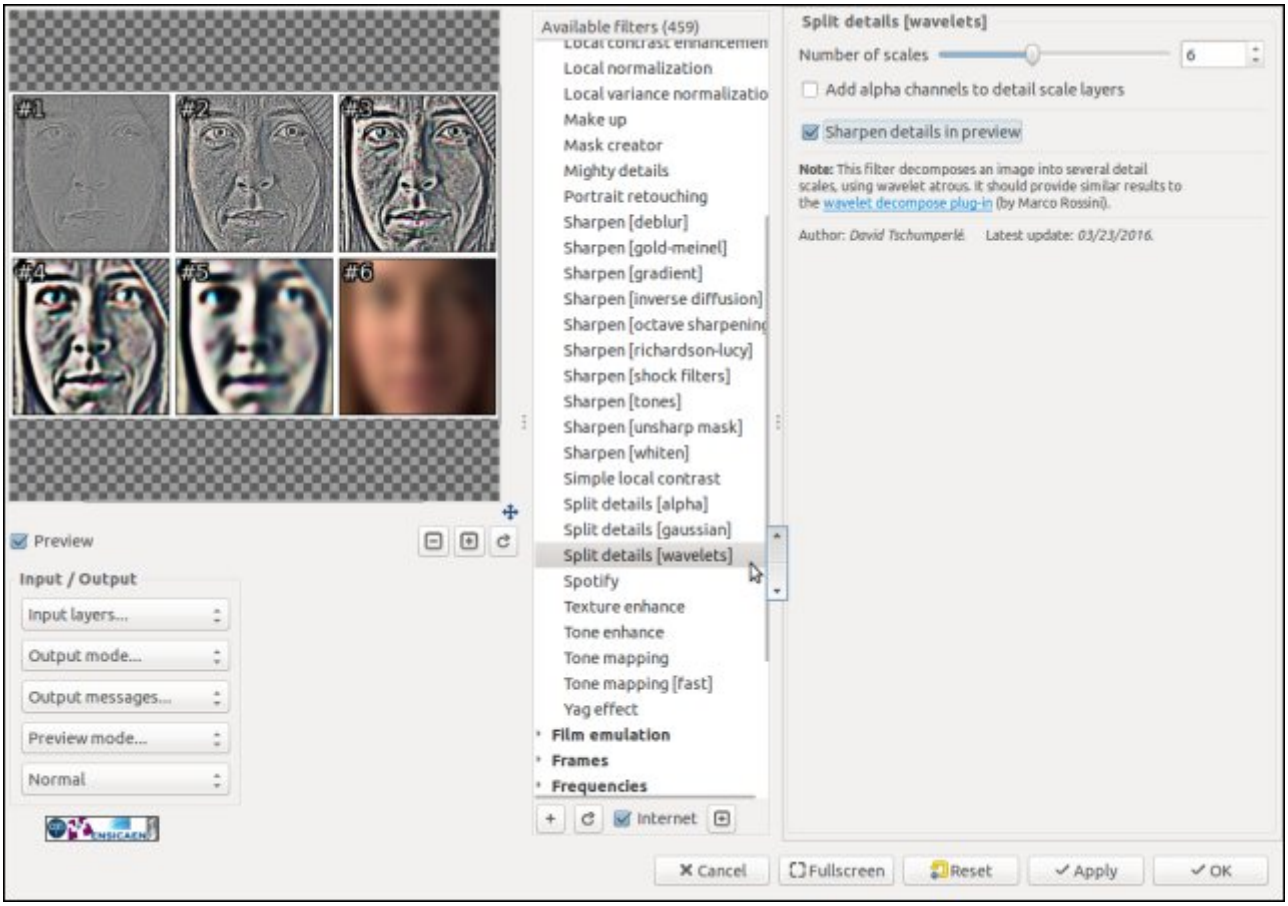

Fig.2.4.1. Aperçu du filtre de décomposition d'image par ondelettes dans le greffon G'MIC pour GIMP.

On peut ainsi travailler sur chaque calque séparément, et ne modifier les détails de l'image que pour une échelle donnée. Il y a de nombreuses applications à ce type de décomposition, l'une des plus spectaculaires étant la possibilité de retoucher la peau dans des photos de portraits : les imperfections de la peau se retrouvent généralement sur les calques correspondant à des échelles de détails *moyens*, alors que la texture naturelle de la peau (les pores) se retrouvent sur les échelles de détails *fins*, et on peut donc sélectivement effacer les imperfections tout en conservant une texture de peau naturelle après retouche (voir l'animation ci-dessous ou encore **ce lien** pour un tutoriel détaillé de la procédure, en utilisant *GIMP*).
Vous avez sans doute déjà vu des photos publicitaires de mannequins ayant une peau exagérément lisse (façon poupée barbie). Dans ce cas, vous savez maintenant que l'infographiste responsable a vraiment fait une retouche de goret ! (le bien-fondé de l'utilité de telles retouches est un autre débat dans lequel on ne se risquera pas ici).

Fig.2.4.2. Un exemple d'utilisation du filtre de décomposition d'image en ondelettes, pour la retouche réaliste de la peau sur un portrait (suppression des imperfections).

2.5. Débruitage d'images par méthode « Patch-PCA »

G'MIC est aussi connu pour posséder de nombreux algorithmes variés de *débruitage* et de *lissage* d'images (plus d'une quinzaine à ce jour). Et bien, il en a maintenant un de plus ! **Repair / Smooth [patch-pca]** est un nouvel algorithme de débruitage d'image performant, basé ***patch*** [W] qui a été ajouté à *G'MIC*. C'est un algorithme parallélisé, mais *très coûteux* en temps de calcul (probablement à éviter sur des machines à moins de 8 coeurs…). En contrepartie, il est capable de débruiter certaines images de façon parfois spectaculaire, en supprimant le bruit et préservant les détails de l'image, comme l'illustre la figure ci-dessous, avec une image contenant un niveau de bruit assez important. (En passant, merci à **Jérome Boulanger** pour ses conseils d'expert sur ce sujet).

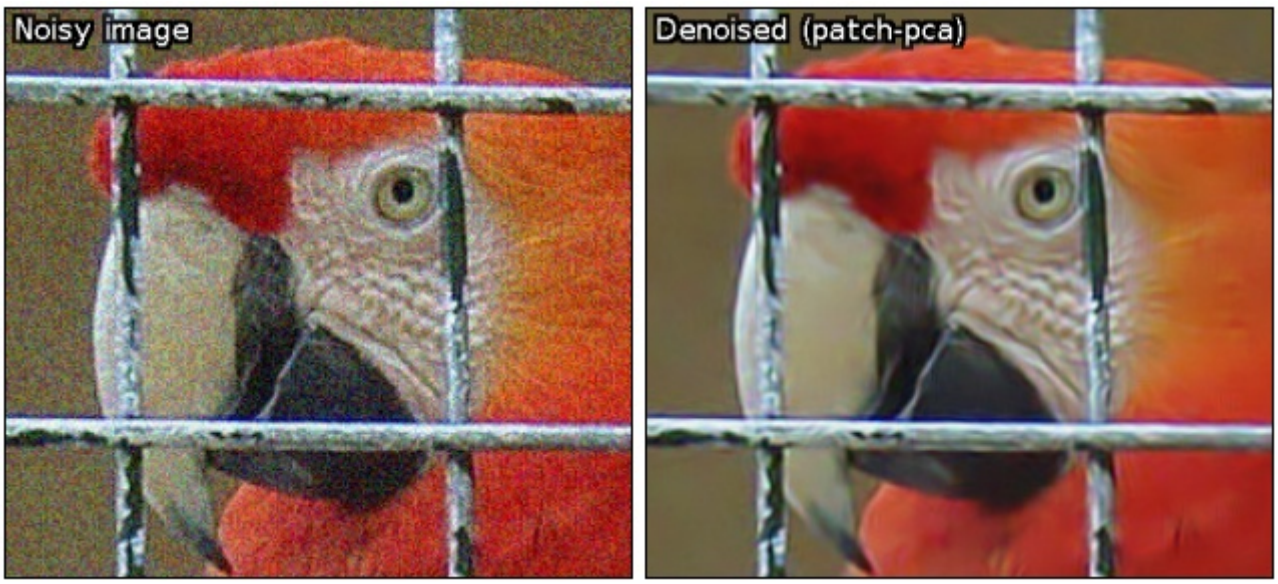

Fig.2.5.1. Résultat du nouvel algorithme de débruitage d'image basé « patch » de G'MIC.

2.6. Effet « Droste » : la mise en abyme continue

L'effet Droste [W] (aussi appelé *mise en abyme*) du nom de la marque de cacao ayant utilisé cet effet dans une de ses publicités, consiste à dessiner une partie de l'image dans elle-même, et ceci de manière récursive. Un filtre **Deformations / Continuous droste** a été récemment ajouté à *G'MIC*, mais n'a en réalité rien de très nouveau puisque c'est « juste » une réécriture complète du **filtre Droste** qui était déjà disponible dans le greffon ***Mathmap*** depuis quelques années. *Mathmap* était un greffon populaire pour *GIMP*, mais il ne semble plus évoluer, ni même maintenu, et l'effet Droste était l'un de ses filtres les plus complexes et les plus emblématiques. *Martin « Souphead »*, un ancien utilisateur de *Mathmap* a donc pris le taureau par les cornes et s'est attelé à la conversion de ce filtre pour *G'MIC*. L'intérêt c'est qu'au passage, l'implémentation devient parallélisée. Pour celles et ceux intéressés par les aspects mathématiques de l'effet *Droste*, on ne peut que recommander la lecture de **cette page didactique** rédigée par un chercheur du *CNRS*, page qui contient des résultats amusants de création de séquences périodiques d'images utilisant cette effet.

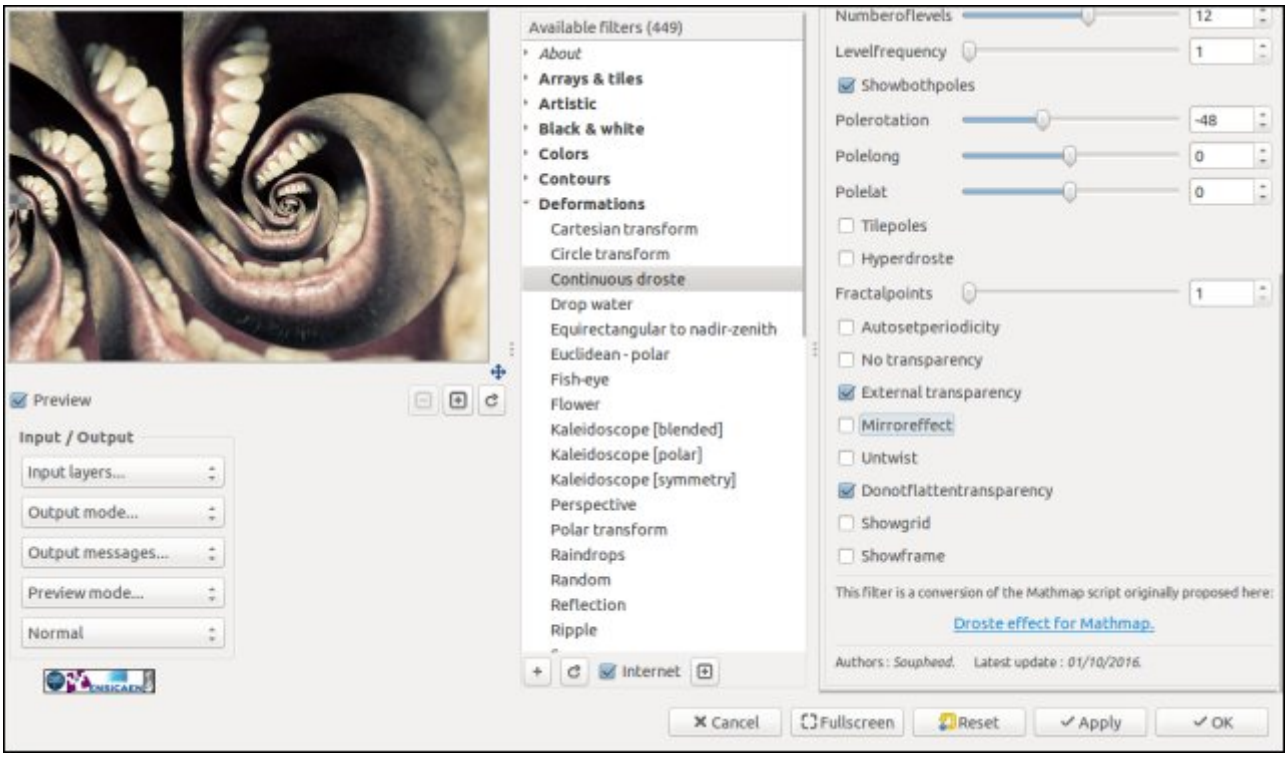

Fig.2.6.1. Aperçu du nouveau filtre « Droste » pour la création d'une mise en abyme, dans le greffon G'MIC pour GIMP.

Avec ce filtre, tous les délires artistiques sont permis. Il est par exemple trivial de créer en quelques clics de souris le résultat présenté dans la figure ci-dessous : il suffit de détourer l'horloge, de rendre le fond transparent, et d'appliquer le filtre *Droste* de *G'MIC*, *et voilà* ! (à ne pas montrer aux gens stressés par le temps qui passe…).

Fig.2.6.2. Exemple de transformation d'image possible avec le filtre « Droste » de G'MIC.

2.7. Transformation équirectangulaire <-> zénith/nadir

Le filtre **Deformations / Equirectangular to nadir-zenith** est également un filtre initialement disponible dans *Mathmap* et qui a été transposé pour *G'MIC*. C'est un filtre utilisé dans le domaine assez restreint du traitement d'images de panoramas utilisant une **projection cylindrique équidistante** [W]. Un tel panorama est en général obtenu comme la fusion de plusieurs photographies prises à des angles différents, la fusion étant effectuée de manière algorithmique (par exemple avec le logiciel libre **Hugin**). Lors de la fusion, il est très fréquent que des pans entiers d'images manquent dans le panorama généré, notamment au niveau des vues de dessus et de dessous (le ***zénith*** [W] et le ***nadir*** [W], voir un exemple dans la figure ci-dessous).

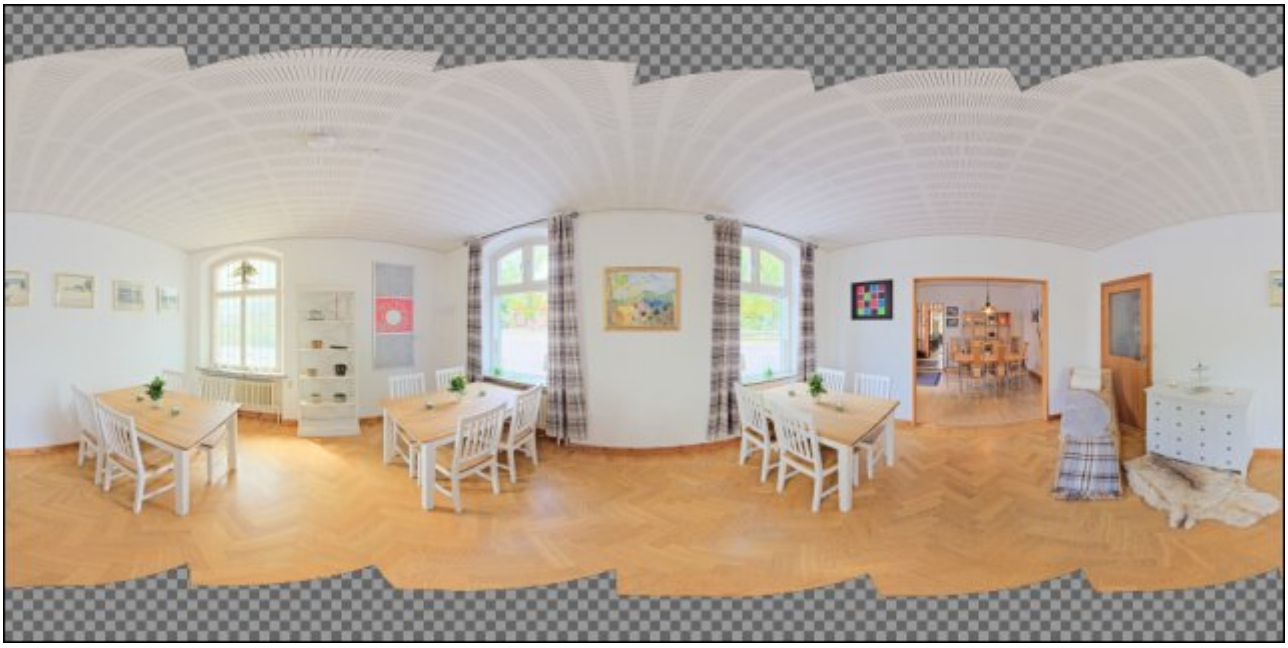

Fig.2.7.1. Panorama obtenu par outil de fusion d'image. Certaines parties de l'image (zénith et nadir) sont manquantes.

Il est souhaitable de pouvoir resynthétiser l'information manquante dans ces zones. Mais comment faire ? La déformation induite par la projection cylindrique équidistante fait que la reconstruction directe est difficile dans ces zones (l'utilisation de l'outil de ***clonage*** n'est pas adapté par exemple). C'est là que le filtre de *G'MIC* intervient, en permettant de recréer des vues *applaties* du zénith et du nadir.

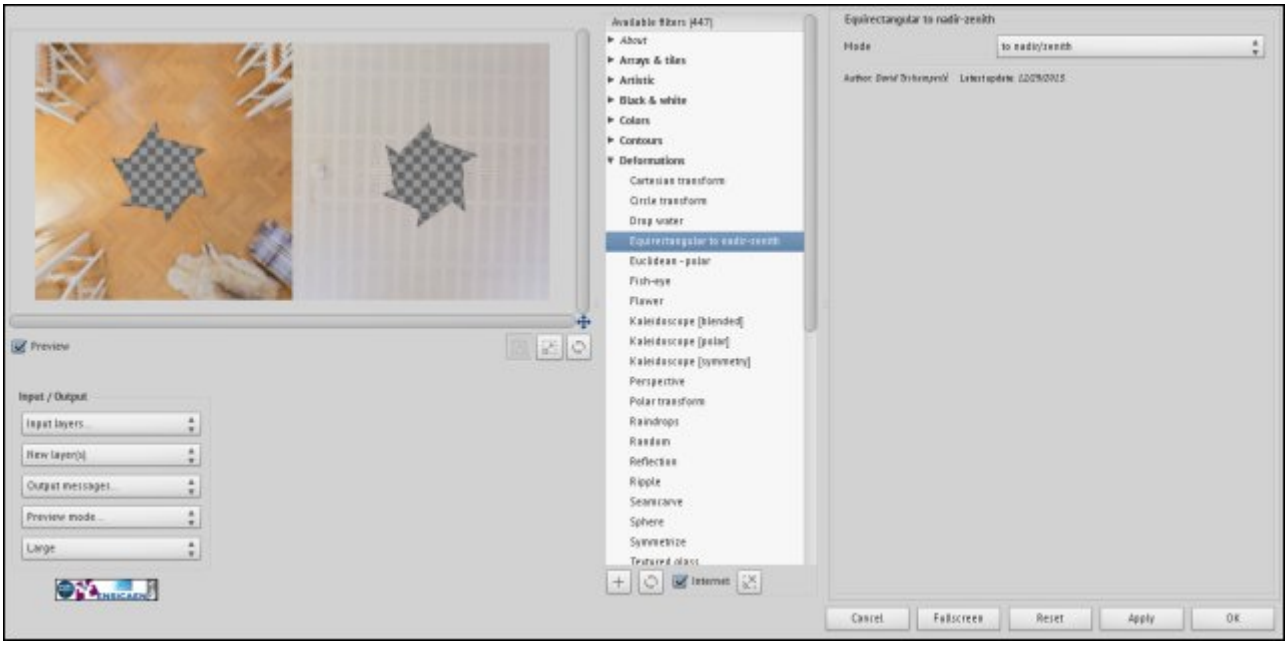

Fig.2.7.2. Récupération des vues applaties du zénith et du nadir de l'image précédente, grâce au filtre G'MIC du greffon pour GIMP.

Une fois ces vues calculées, il devient plus facile de boucher les trous, en utilisant par exemple un filtre de reconstruction de type *Inpainting* ou l'outil de clonage si on préfère faire ça manuellement.

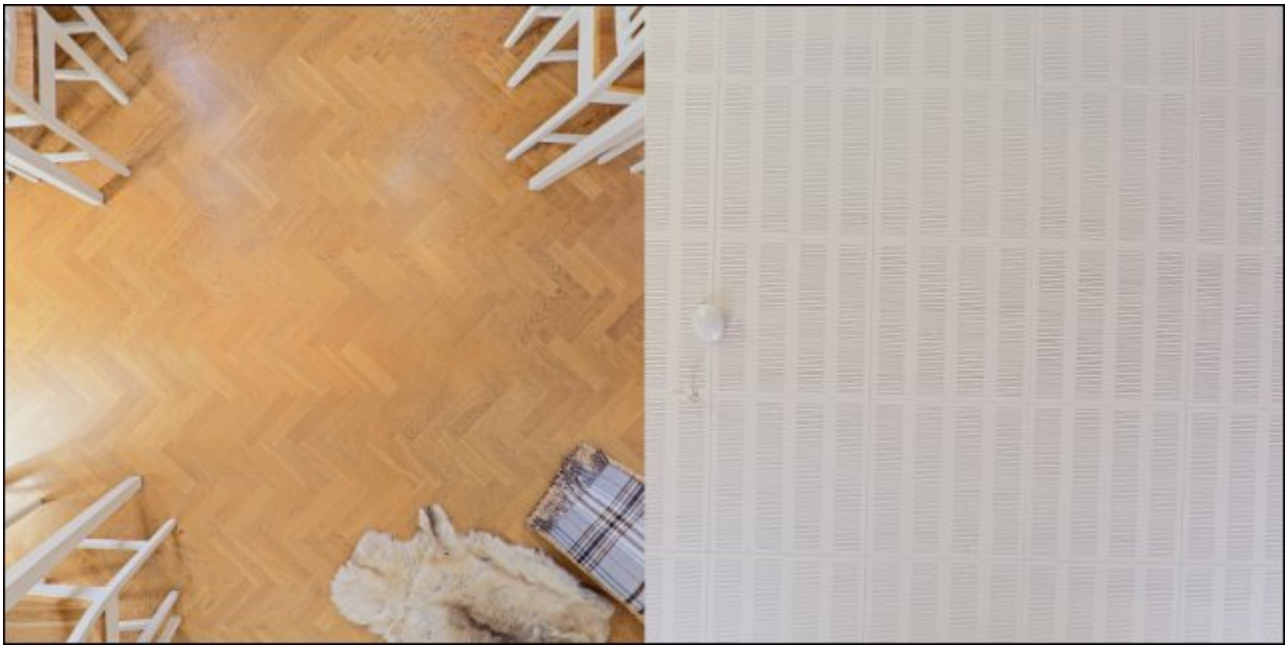

Fig.2.7.3. Rebouchage des trous, par une technique quelconque (« inpainting » ou outil de « clonage » par exemple).

Il suffit ensuite d'invoquer ce même filtre, en inversant cette fois la transformation, et de réinsérer les zénith/nadir reconstruits dans l'image panorama originale, et le tour est joué. On obtient une belle image panorama complète (voir figure ci-dessous). Notez comme la déformation de l'image est importante dans ces zones, et comment il aurait été difficile de reboucher les trous en agissant directement sur l'image du panorama original.

Fig.2.7.4. Application de la transformation inverse, et insertion dans le panorama d'origine.

Les images présentées dans cette section ont été aimablement fournies par ***Morgan Hardwood***. *Morgan* a d'ailleurs écrit un tutoriel détaillé sur cette technique de rebouchage d'images de panoramas, qui est **consultable ici**.

3. Autres améliorations et faits notables

Pour finir, voici en vrac quelques points marquants concernant le développement du projet *G'MIC* :

- Le filtre **Rendering / Kitaoka Spin Illusion** est une autre conversion d'un filtre *Mathmap* réalisé par *Martin « Souphead »*. Il permet de générer un certain type d'**illusions d'optiques** comme le montre la figure ci-

dessous (si vous êtes épileptiques, un conseil, fermez les yeux !)

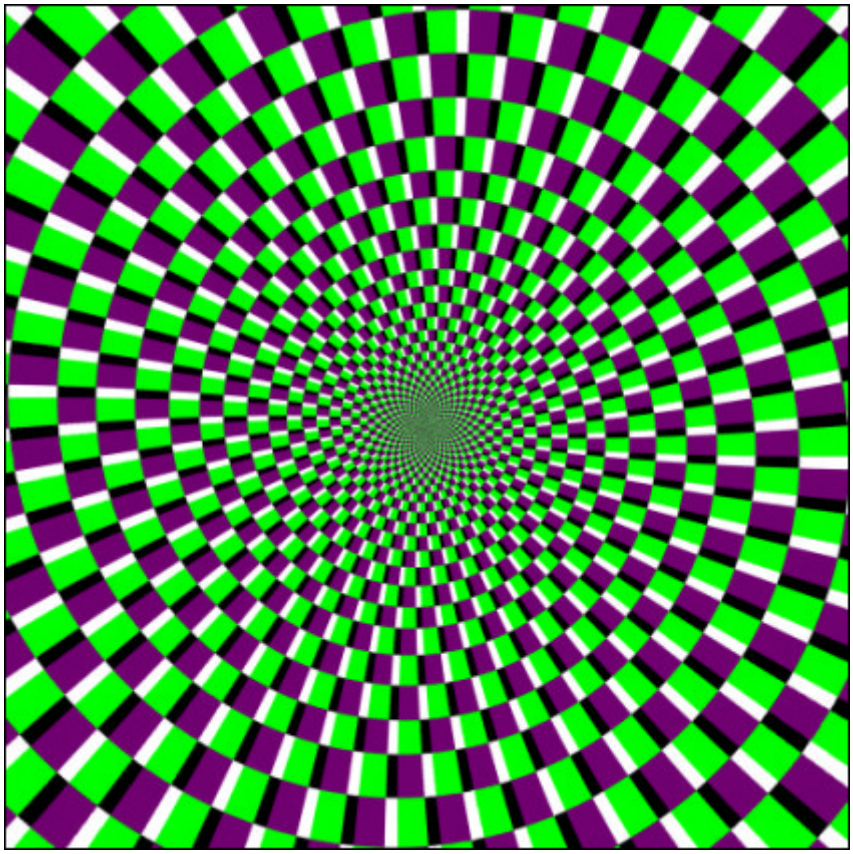

Fig.3.1. Résultat du filtre « Kitaoka Spin Illusion ».

- Le filtre **Colors / Color blindness** transforme une image en simulant différents types de **daltonisme** [W]. Ce filtre peut être utile pour tester l'accessibilité de sites ou de documents graphiques aux daltoniens. Nous avons repris les transformations colorimétriques dont le lien apparaît sur le site ***Coblis***, site qui propose également ce genre de simulation, en ligne. Les résultats obtenus avec le filtre *G'MIC* sont donc à priori strictement identiques, mais peuvent s'effectuer facilement sur des lots d'images par exemple.

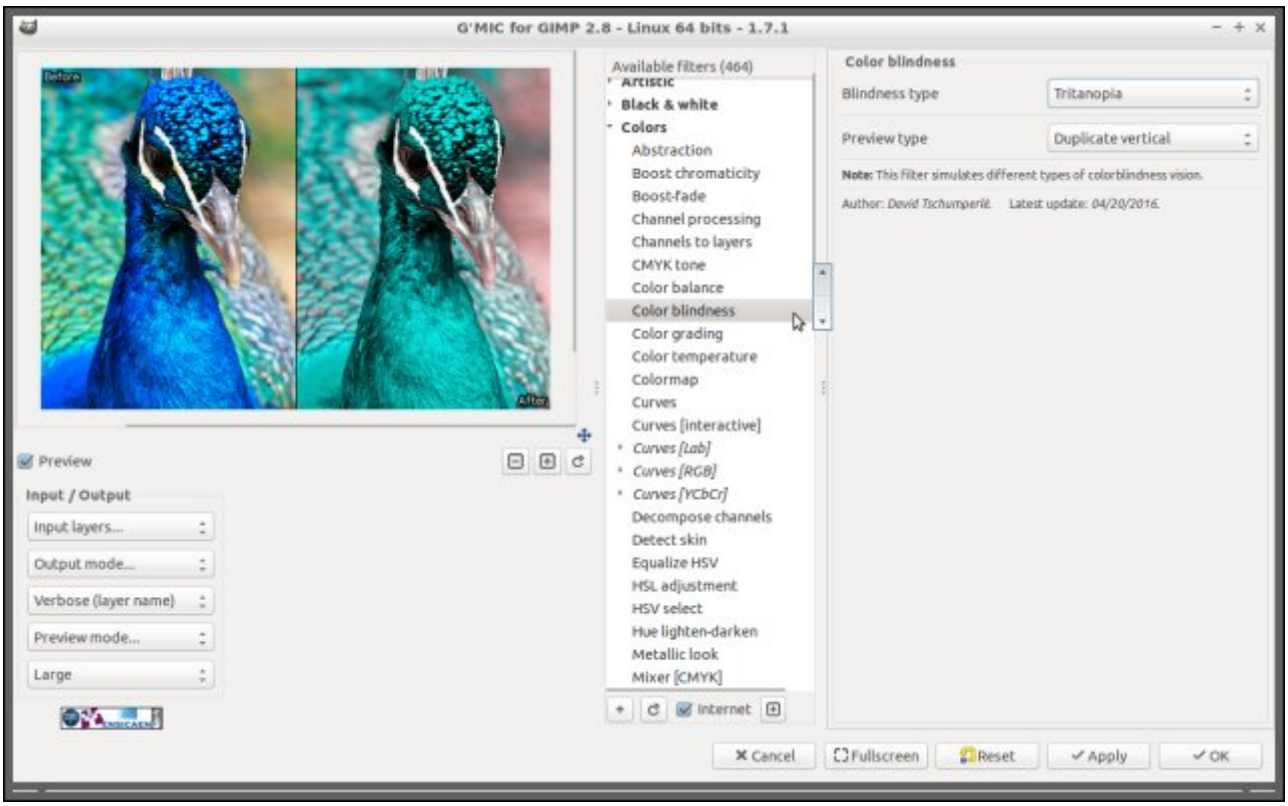

Fig.3.2. Aperçu du filtre de simulation de différents types de daltonisme dans le greffon G'MIC pour GIMP.

- Depuis quelques années maintenant, *G'MIC* intègre un évaluateur d'expressions mathématiques, très commode pour réaliser des calculs lors de l'application de filtres (nous en avions d'ailleurs déjà longuement parlé **lors de la dépêche précédente**). Cet évaluateur d'expression se dote de nouvelles fonctionnalités intéressantes, en particulier la possibilité de faire du calcul avec des variables de type complexe, vectoriel ou matriciel, mais aussi de créer ses propres fonctions mathématiques personnalisées. Par exemple, l'implémentation classique du rendu de **l'ensemble de *Mandelbrot*[W]**, réalisé en estimant la convergence d'une suite complexe, peut s'écrire directement de la façon suivante en ligne de commande:

```
$ gmic 512,512,1,1,"c = 2.4*[x/w,y/h] - [1.8,1.2]; z = [0,0]; for (iter = 0, cabs(z)<=2 && ++iter<256,
```

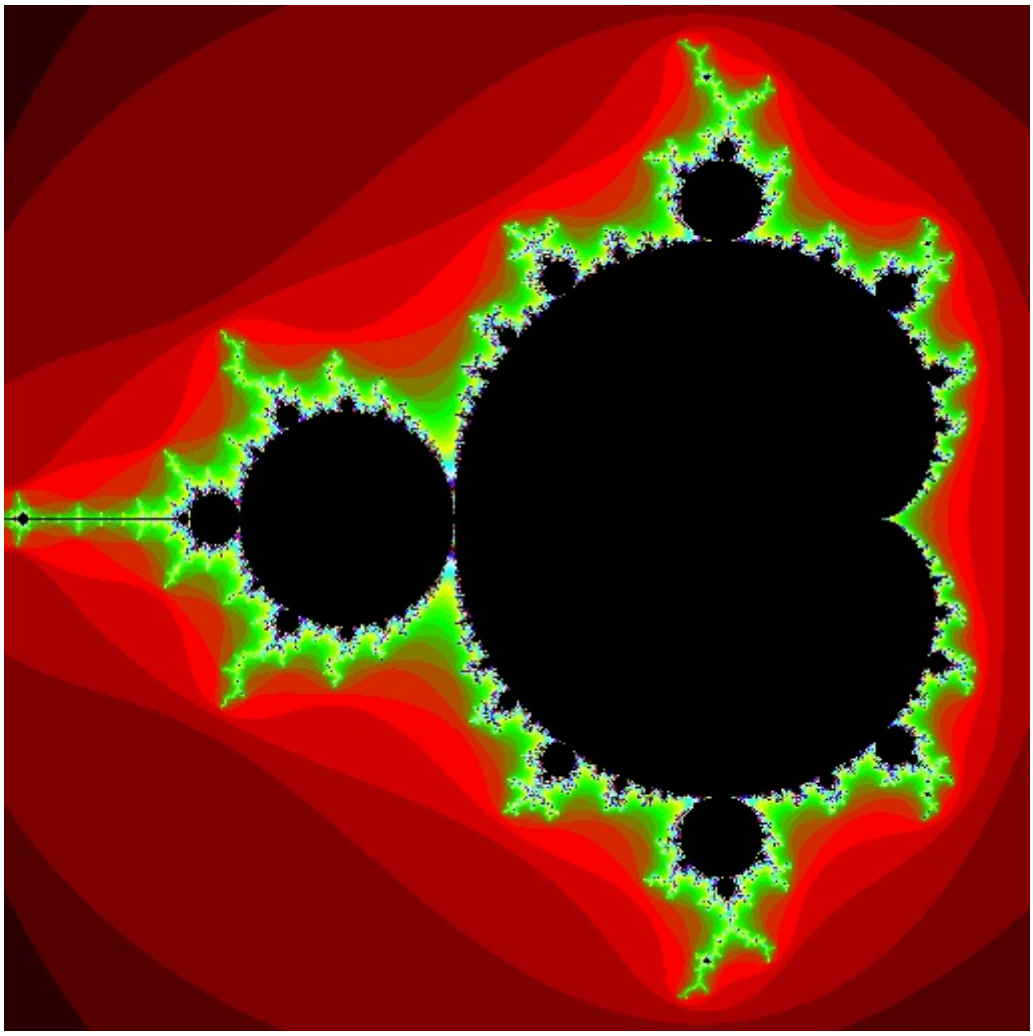

Fig.3.3. Utilisation de l'évaluateur d'expression mathématiques de G'MIC pour calculer un rendu de l'ensemble de Mandelbrot.

Les possibilités de calcul en sont grandement développée, puisque l'on n'est plus limité à l'utilisation de variables scalaires, mais qu'on peut définir des filtres qui, pour chaque pixel d'une image d'entrée, vont pouvoir effectuer rapidement des résolutions de systèmes linéaires ou encore des décompositions en valeurs propres/vecteurs propres. C'est un peu comme si on disposait d'un mini-(mini)-***Octave*** à l'intérieur de *G'MIC*. Le filtre *Brushify* décrit plus haut utilise d'ailleurs de manière intensive ces nouvelles possibilités. À noter que cet évaluateur d'expression possède son propre ***JIT*[W]** pour accélérer le calcul d'une expression lorsqu'elle est réalisée sur plusieurs milliers de valeurs simultanément.

- Une autre contribution technique importante a été apportée par ***Tobias Fleischer***, avec la création d'une ***API*[W]** *C* pour appeler les fonctions de la bibliothèque ***libgmic*** (bibliothèque des fonctionnalités de *G'MIC* possédant initialement une *API C++*). Comme l'***ABI*[W]** *C* est standardisée (contrairement à celle du *C++*),

G'MIC peut donc plus facilement s'interfaçer avec d'autres langages que le *C++*. On peut par exemple imaginer dans le futur la création d'*APIs G'MIC* pour des langages, comme *Python*. En passant : si quelqu'un est motivé pour réaliser ce genre de choses, qu'il n'hésite surtout pas à nous contacter ! *Tobias* utilise actuellement cette nouvelle *API C* pour développer des greffons basés sur *G'MIC* respectant l'*API* ***OpenFX***[W]. Ces greffons devraient donc être utilisables indifféremment dans des logiciels d'édition vidéo comme **After effects**[W], **Sony Vegas Pro**[W] ou encore **Natron** (voir figure ci-dessous). C'est un travail qui est toujours en cours.

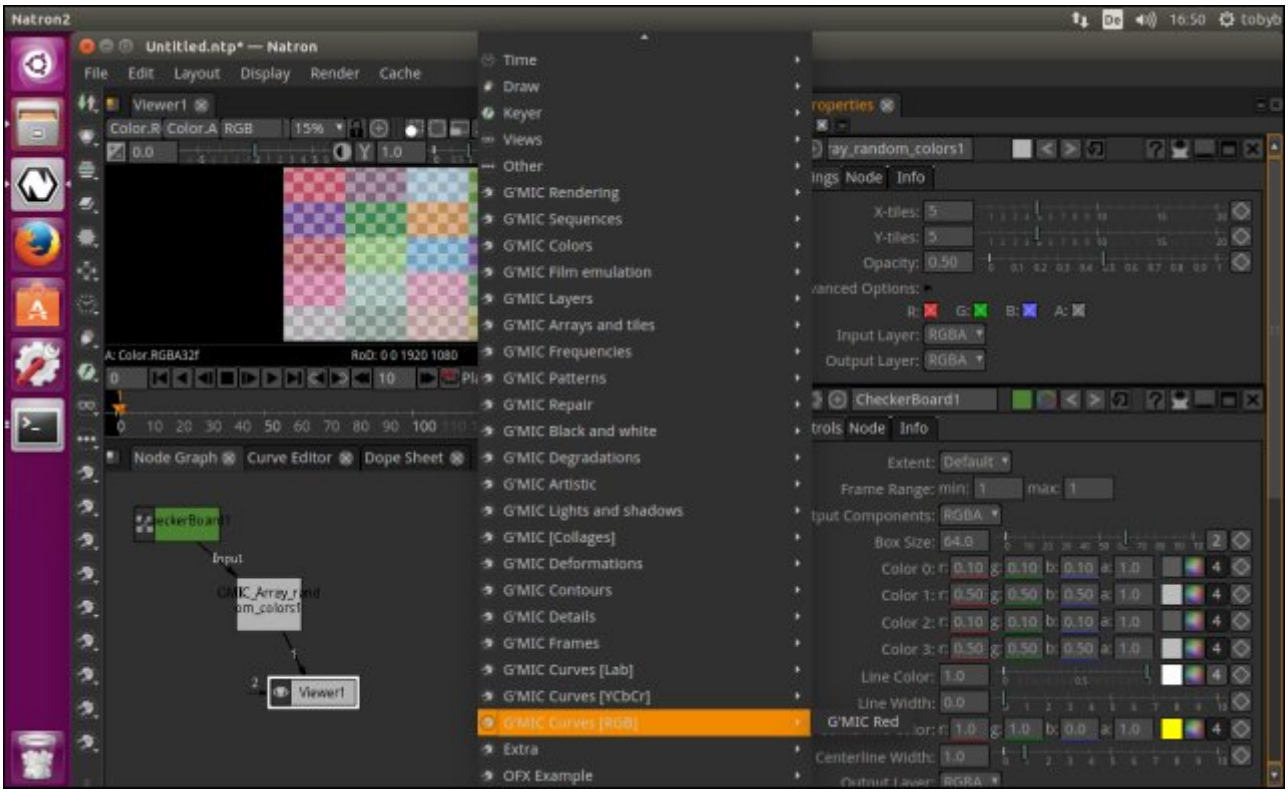

Fig.3.3. Aperçu des greffons OpenFX basés sur G'MIC, tournant sous Natron.

- Un autre contributeur ***Robin « Starfall Robles »*** a initié le développement d'un **script Python** permettant **d'intégrer des fonctionnalités de *G'MIC* dans *Blender*** (plus précisément dans son éditeur de séquence vidéo *VSE*). Ce script très récent, toujours en cours de développement, permet déjà d'appliquer différents effets *G'MIC* sur des séquences d'images, comme vous pouvez le voir sur la figure ci-dessous (et sur **cette vidéo de démonstration**).

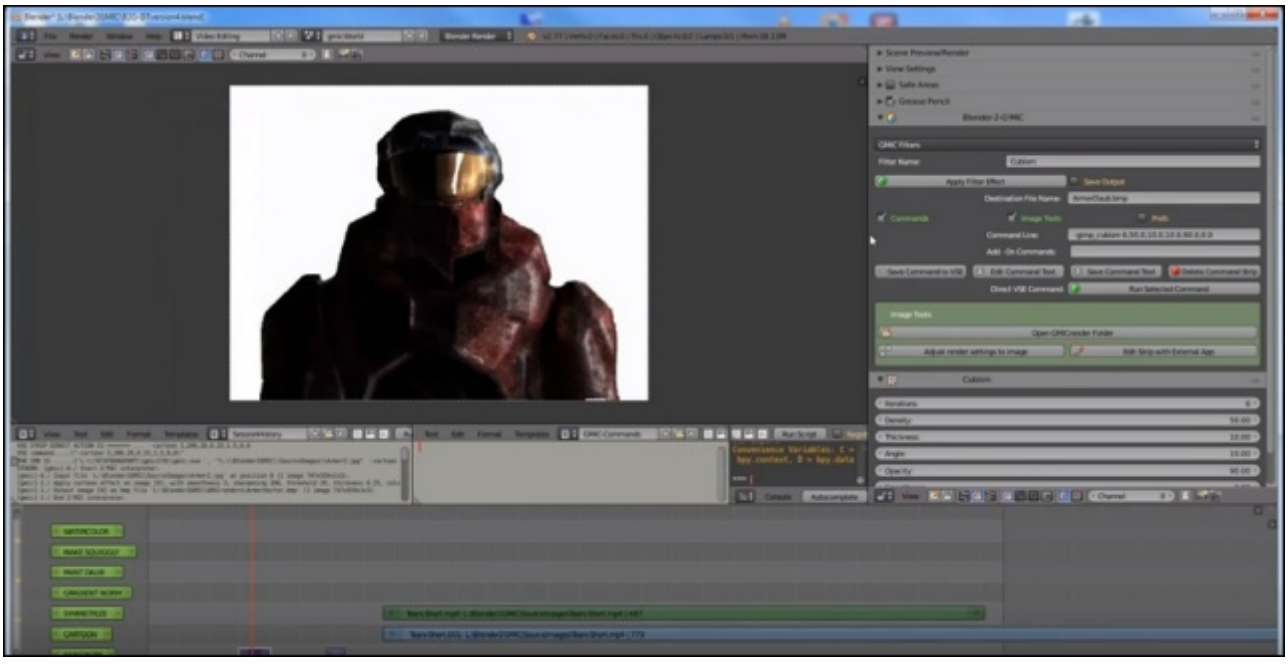

Fig.3.4. Aperçu du script d'interfaçage G'MIC fonctionnant dans le VSE de Blender.

- Certaines fonctionnalités de *G'MIC* ont également fait leur apparition dans le logiciel de montage vidéo non-linéaire ***Flowblade***, grâce au travail acharné de ***Janne Liljeblad*** son programmeur principal (voir figure ci-dessous). Là encore, le but est de permettre d'appliquer des effets *G'MIC* sur des séquences d'images dans un but essentiellement artistique, comme le montre **cette vidéo**, ou encore **celle-ci**.

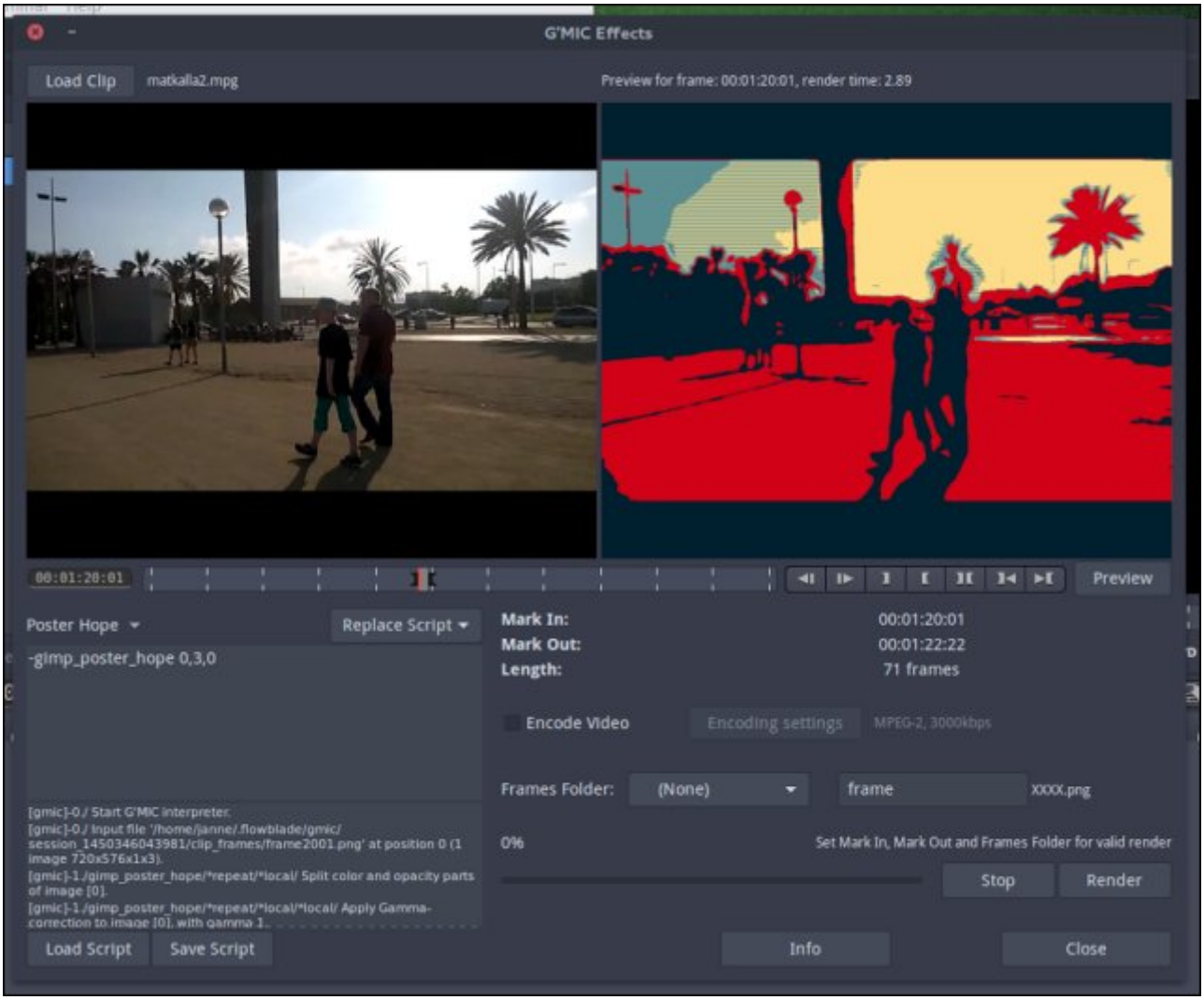

Fig.3.5. Aperçu d'un filtre G'MIC tournant sous Flowblade, éditeur non-linéaire de vidéos.

- Notons également que de plus en plus de ressources extérieures faisant mention de *G'MIC* font leur apparition sur la toile : des tutoriaux et des articles de blog (**ici**, **ici**, **ici**…), ou encore des vidéos de démonstrations (**ici**, **ici**, **ici**, **ici**…). C'est très positif pour la visibilité du projet, et en même temps cela fait vraiment plaisir à voir. Merci donc à tout ces gens qui prennent le temps d'en parler de manière bénévole et désintéressée !

4. Comment tout cela va évoluer ?

Quelques petites observations :

- Comme vous pouvez le constater, le développement du projet *G'MIC* se poursuit à un rythme soutenu. Ses fonctionnalités semblent intéresser de plus en plus d'utilisateurs (ce que confirme les statistiques de visites/téléchargements du site web), mais aussi de plus en plus de développeurs : aujourd'hui, on retrouve des intégrations ou des greffons (plus ou moins aboutis) basés sur *G'MIC* dans des logiciels libres aussi divers que ***GIMP***, ***Krita***, ***Blender***, ***Photoflow***, ***Flowblade***, **Veejay**, ***EKD*** et dans un futur proche (du moins on l'espère) ***Natron***.

- L'un ne va probablement pas sans l'autre : le fait d'avoir des utilisateurs nous encourage à ajouter de nouvelles fonctionnalités régulièrement, ce qui attire aussi de nouveaux utilisateurs. Tant que ça fonctionne de cette façon, on essayera de continuer ! Car notons tout de même que tout ceci demande pas mal de temps (pour ma part, entre *10* et *15* heures par semaine *en dehors* de mes heures officielles de travail).
- Tout ça pour redire un grand merci aux utilisateurs et aux contributeurs (toujours plus nombreux), aux curieux et à ceux qui font de la publicité au projet directement ou indirectement. Ça aide énormément !

En réalité, on ne sait pas encore comment le projet *G'MIC* va évoluer dans le futur, mais il y a déjà tellement de choses à faire dans le présent qu'on se concentre dessus pour le moment. On vous donne peut-être rendez vous dans quelques mois pour la suite des aventures de *G'MIC*. On vous invite également à rejoindre la communauté présente sur notre forum officiel sur **pixls.us** pour obtenir plus de renseignements et répondre à vos questions sur le projet. Et surtout, en attendant, n'hésitez pas à vous mettre au traitement d'images **libre** !

Aller plus loin

- **Le projet G'MIC** (520 clics)
- **Le greffon G'MIC pour GIMP** (374 clics)
- **Initiation à G'MIC en ligne de commande** (153 clics)
- **Documentation technique de référence** (140 clics)
- **Série d'articles G'MIC sur Linuxfr** (263 clics)

G'MIC dans Blender

Posté par **Xavier Claude** le 05/05/16 à 11:07. Évalué à 5.

> La vidéo d'exemple de G'MIC dans Blender ne montre que l'application de filtre sur une image statique, est-ce aussi possible de le faire sur des vidéos ?
>
> --
> « Rappelez-vous toujours que si la Gestapo avait les moyens de vous faire parler, les politiciens ont, eux, les moyens de vous faire taire. » Coluche

Re: G'MIC dans Blender

Posté par **David Tschumperlé** (**site web personnel**) le 05/05/16 à 12:08. Évalué à 4.

> Oui c'est le but. J'avoue que je n'ai pas encore essayé, je ne maitrise pas du tout Blender.

Re: G'MIC dans Blender

Posté par **Ichardon** le 06/05/16 à 09:30. Évalué à 3.

> Oui apparemment c'est possible de le faire sur des vidéos : voir cette **autre vidéo youtube** de l'auteur

Merci

Posté par **Zatalyz** (**site web personnel**, **adresse XMPP**) le 05/05/16 à 12:38. Évalué à 6.

> Super travail, merci !
>
> Ça devient vraiment trop facile de transformer ses images avec tous ces filtres !