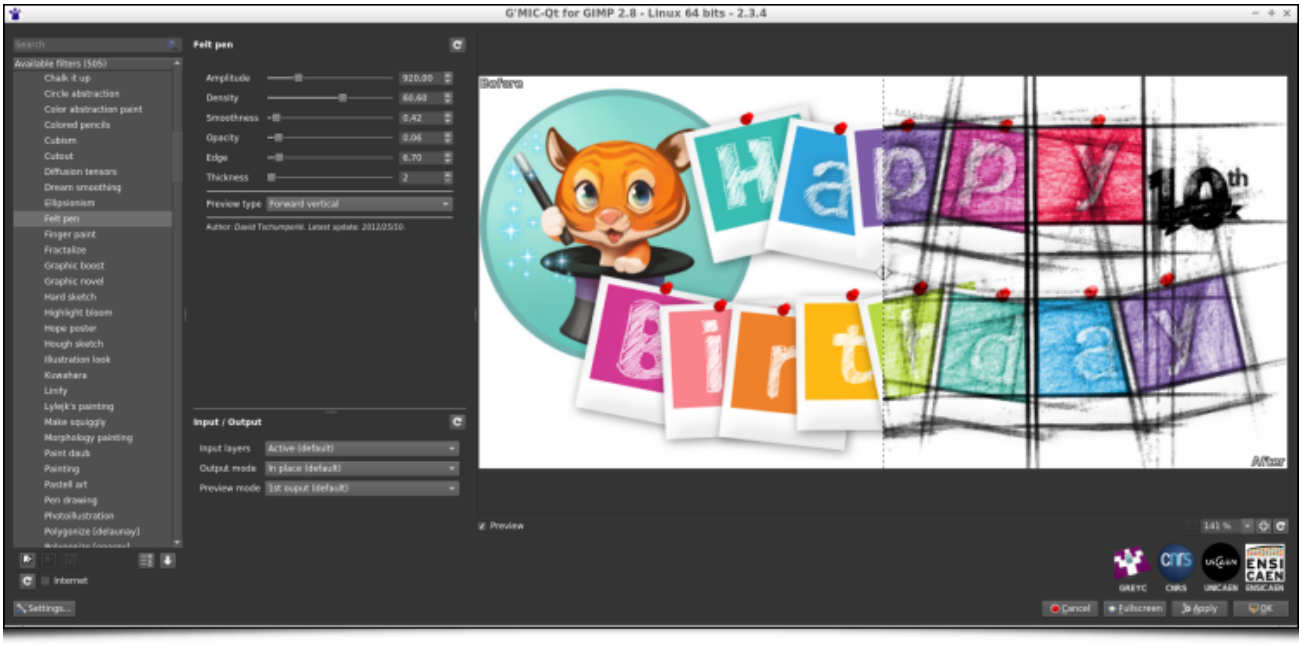
G’MIC 2.3.4 : traiter ses images, en se disant « déjà 10 ans ! »

Posté par David Tschumperlé (site web personnel) le 21/08/18 à 08:58. Édité par Davy Defaud, Benoît Sibaud et Xavier Teyssier. Modéré par Xavier Teyssier. Licence CC By-SA.

Étiquettes : greyc , g'mic , traitement_d'images + Étiqueter

L’équipe IMAGE du GREYC est heureuse de pouvoir fêter avec vous les dix années d’existence du logiciel *G’MIC*, son cadriciel libre (sous licence CeCILL), générique et extensible pour le traitement des images [W].
Le GREYC est un laboratoire de recherche publique en sciences du Numérique, situé à Caen en Normandie, et chapeauté par trois tutelles : le CNRS (UMR 6072), l’Université de Caen et l’ENSICAEN.



G’MIC-Qt, l’interface utilisateur principale du projet libre G’MIC.

Cet anniversaire décennal nous donne l’occasion rêvée, d’une part, d’annoncer la sortie d’une nouvelle version de ce logiciel libre (numérotée 2.3.4), et d’autre part, de partager avec vous (comme à notre habitude) un résumé des dernières fonctionnalités notables ajoutées depuis la dernière dépêche sur G’MIC, publiée sur *LinuxFr.org* en février 2018.

Sommaire

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

1. Retour sur dix années de développement

G'MIC est un logiciel multi-plate-forme (GNU/Linux, macOS, Windows…) fournissant différentes interfaces utilisateur pour la manipulation de données image *génériques*, à savoir des images ou des séquences d'images hyperspectrales 2D ou 3D à valeurs flottantes (incluant donc les images couleur « classiques »). Plus de **mille opérateurs** différents de traitement d'images sont inclus, nombre extensible à l'envi puisque les utilisateurs peuvent ajouter leurs propres fonctionnalités via l'utilisation d'un langage de script intégré.

C'est fin juillet 2008, que les premières lignes de *G'MIC* sont rédigées (en *C++*).
À l'époque, j'étais le principal développeur impliqué dans ***CImg***, une bibliothèque *C++ open source* légère pour le traitement d'images, et je réalisais le constat suivant :

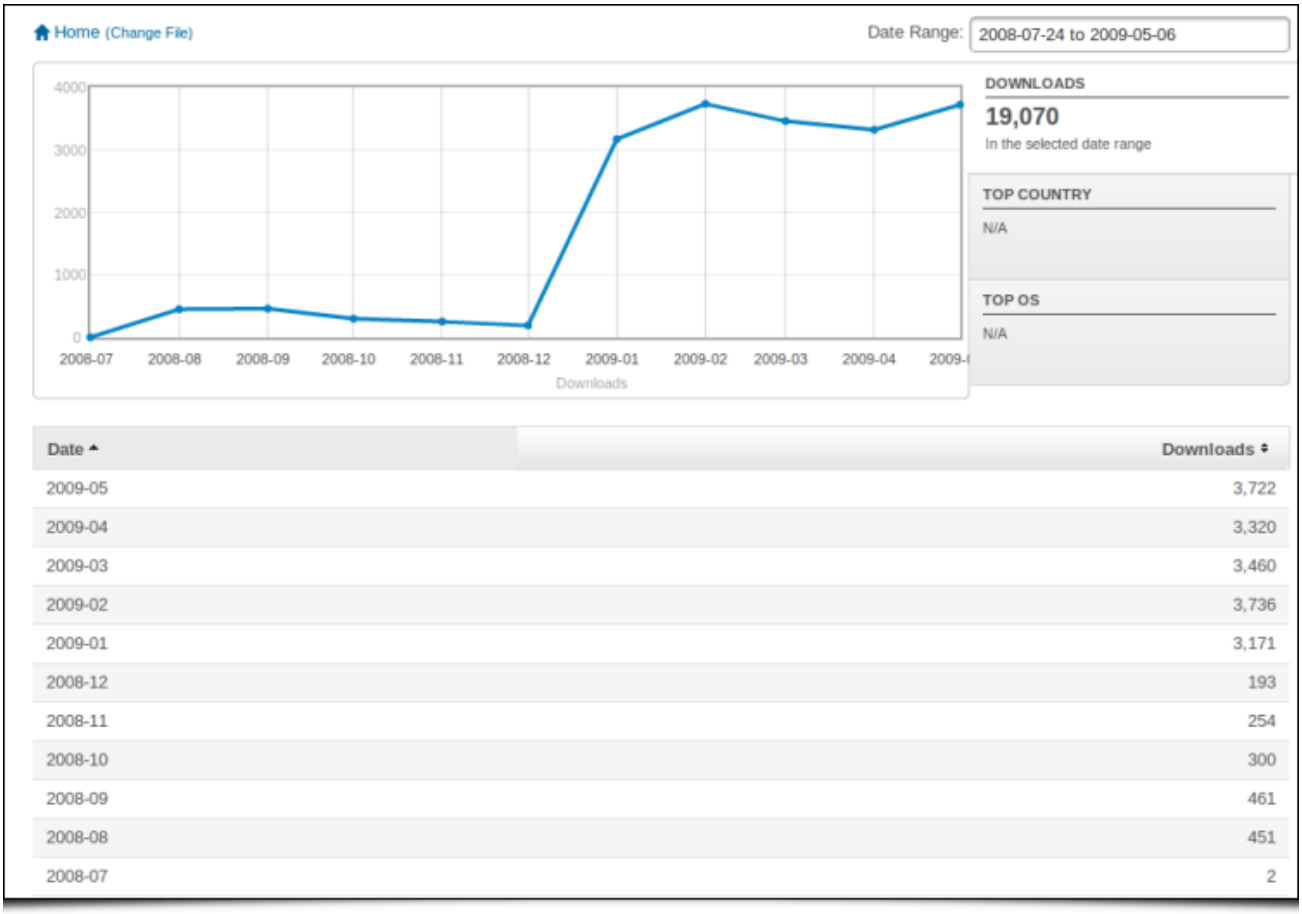
- l'objectif initial de *CImg*, qui était de proposer une bibliothèque « minimale » de fonctionnalités pour aider les développeurs *C++* à élaborer des algorithmes autour du traitement d'images, était globalement atteint :
  - la plupart des algorithmes que je considérais comme « essentiels » en traitement d'images y étaient intégrés,
  - *CImg* était initialement conçue pour rester légère, et je ne souhaitais donc pas y inclure *ad vitam æternam* de nouveaux algorithmes, qui seraient trop lourds ou trop spécifiques et qui trahiraient le concept initial de la bibliothèque ;
- cette satisfaction faisait néanmoins place à une certaine déception ; quel dommage de n'avoir pu toucher qu'un public finalement assez restreint, possédant à la fois des connaissances en *C++* **et** en traitement d'images ! Une des évolutions naturelles du projet, consistant à créer des bibliothèques de liaison (***bindings***[W]) de *CImg* pour d'autres langages de programmation, ne m'ouvrait pas de perspectives très réjouissantes, du point de vue de l'intérêt que j'y trouvais en développement informatique ; sans compter que ces *bindings* potentiels ne concernerait, là encore, qu'un public ayant une expertise en développement.

Mon envie prenait donc progressivement forme : il fallait proposer un moyen d'utiliser les fonctionnalités de traitement d'images de *CImg* pour les **non-programmeurs**. Et pourquoi pas, dans un premier temps, en élaborant un outil utilisable en ligne de commande (à la façon du fameux ***convert*** d'***ImageMagick***) ? Une première tentative, en juin 2008 (*inrcast*, qui avait été présentée dans **un journal *LinuxFr***), se révéla infructueuse mais me permit de mieux cerner les spécificités que se devait de posséder ce genre d'outils, pour traiter confortablement des images en ligne de commande.
Il m'apparut en particulier que la **concision** et la **cohérence** de la syntaxe commandant l'outil devaient être les deux piliers principaux sur lesquels il fallait se reposer. Ces aspects sont ceux qui m'ont demandé le plus d'efforts en recherche et développement (les fonctionnalités de traitement d'images proprement dites étant déjà implémentées dans *CImg*). En fin de compte, cela m'amènera bien plus loin que ce qui était prévu initialement, puisque *G'MIC* se dotera successivement d'un **interpréteur**[W] de **son propre langage de script**, puis d'un compilateur à la volée (***JIT***[W]) pour l'évaluation d'expressions mathématiques et d'algorithmes de traitement d'images travaillant au niveau pixel.

Fin juillet 2008, je me mettais donc au travail avec les idées (presque) claires, et étais heureux d'annoncer ici même, quelques jours plus tard, la sortie d'**une première ébauche de *G'MIC***. Le projet était officiellement en marche !

*Fig. 1.1 : Logo du projet G'MIC, cadriciel libre pour le traitement d'images, et sa mignonne petite mascotte « Gmicky » (illustrée par **David Revoy**).*

Quelques mois plus tard, en janvier 2009, enrichi par mon expérience précédente de développement du logiciel ***GREYCstoration*** (outil libre pour le débruitage et l'interpolation non linéaire d'images, dont un greffon existait pour ***GIMP***), et dans l'espoir de toucher un public encore plus large, je diffusais une version de *G'MIC* se déclinant sous forme **d'un greffon *GTK* pour *GIMP***.
Cette étape s'est avérée déterminante pour le projet *G'MIC*, le faisant passer de hautement confidentiel à doucement populaire :), comme l'illustre le saut significatif visible dans les statistiques de téléchargements mensuels de l'époque, présentées ci-dessous (le projet était alors hébergé sur ***Sourceforge***).



Date ▲	Downloads ⇕
2009-05	3,722
2009-04	3,320
2009-03	3,460
2009-02	3,736
2009-01	3,171
2008-12	193
2008-11	254
2008-10	300
2008-09	461
2008-08	451
2008-07	2

Fig. 1.2 : Statistiques de téléchargement mensuels de G'MIC, entre juillet 2008 et mai 2009 (arrivée du greffon pour GIMP en janvier 2009).

Cet intérêt soudain pour le greffon de la part d'utilisateurs différents de GIMP (photographes, illustrateurs et autres artistes en tout genre…) fut en effet une vraie rampe de lancement pour le projet, avec l'apparition rapide de contributions et suggestions extérieures diverses et variées (pour le code, la gestion des forums, des pages Web, l'écriture de tutoriels, la réalisation de vidéos…). L'effet communautaire bénéfique du logiciel libre, souvent idéalisé, survenait finalement assez rapidement ! Avec aussi pour conséquence d'amener certains utilisateurs-développeurs à s'intéresser plus en détails au fonctionnement de l'interface originelle en ligne de commande et à son lan-

gage de script associé (qui n'intéressait pas grand monde jusque-là, il faut bien l'avouer !). De là, plusieurs d'entre eux **franchirent le pas** et commencèrent à élaborer et implémenter de nouveaux filtres de traitement d'images en langage *G'MIC*, intégrés progressivement au greffon pour GIMP (aujourd'hui, ces contributions représentent quasiment la moitié des filtres disponibles dans le greffon).

En parallèle, les apports importants et répétés de ***Sébastien Fourey***, collègue de l'équipe *IMAGE* du *GREYC* (et développeur *C++* chevronné s'il en est) ont permis d'améliorer significativement le confort d'utilisation de *G'MIC*. *Sébastien* est en effet à l'origine du développement des interfaces graphiques principales du projet, à savoir :

- le service Web ***G'MIC Online*** (qui a plus tard été réorganisé par le service « développement » du *GREYC*) ;
- Le logiciel libre ***ZArt***, une interface graphique, basé sur la bibliothèque ***Qt***, pour l'application de filtres *G'MIC* sur des séquences vidéos (provenant de fichiers ou de flux de caméras numériques) ;
- et surtout, Sébastien s'est attaqué, fin 2016, à la réécriture complète du greffon *G'MIC* pour GIMP sous une forme plus **générique**, dénommée ***G'MIC-Qt*** ; ce composant, basé également sur la bibliothèque Qt (comme son nom l'indique), se présente sous la forme d'un greffon unique qui fonctionne de manière équivalente sous **GIMP** et **Krita**, deux des logiciels libres de référence pour la retouche photographique et la peinture numérique. *G'MIC-Qt* a aujourd'hui complètement supplanté le greffon GTK d'origine grâce à ses nombreuses fonctionnalités : moteur de recherche de filtres intégré, meilleure prévisualisation, interactivité supérieure, etc. C'est aujourd'hui l'interface la plus aboutie et la plus utilisée du projet *G'MIC*, et nous espérons d'ailleurs pouvoir la décliner dans le futur pour d'autres logiciels hôtes (contactez-nous si vous êtes intéressés par ce sujet !).

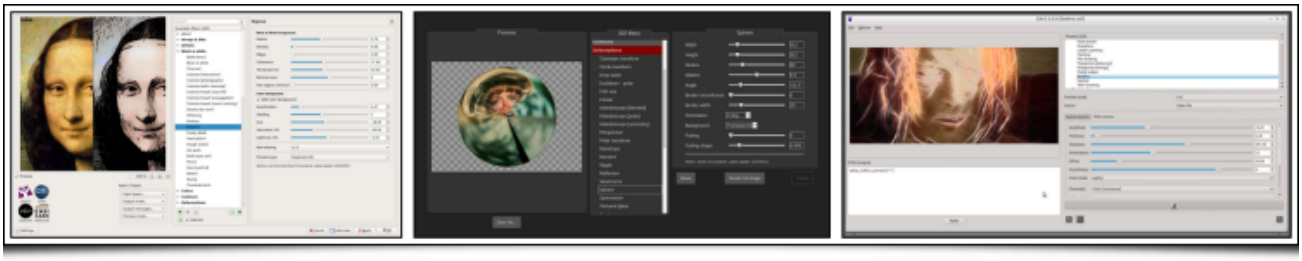


Fig. 1.3 : Différentes interfaces graphiques du projet G'MIC, développées par Sébastien Fourey : G'MIC-Qt, G'MIC Online et ZArt.

L'idée de cette dépêche n'étant pas de rentrer trop en détails dans l'historique du projet, affirmons simplement que l'on n'a pas vraiment eu le temps de s'ennuyer ces dix dernières années !
Aujourd'hui, Sébastien et moi-même sommes les deux mainteneurs principaux du projet *G'MIC* (Sébastien, majoritairement pour ce qui concerne les aspects « interface », et moi-même pour le développement et l'amélioration des filtres et du cœur de calcul), ceci, en complément de notre activité professionnelle principale (la recherche, l'enseignement et l'encadrement).

Avouons-le, gérer un projet libre comme *G'MIC* prend un temps considérable, malgré sa taille modeste (≈ 120 000 lignes de code). Mais l'objectif de départ a été atteint : des milliers d'utilisateurs non-programmeurs ont l'occasion d'utiliser librement et aisément nos algorithmes de traitement d'images, et ce, dans de nombreux domaines différents : **retouche photographique** [W], **illustration et peinture numérique** [W], **traitement vidéo** [W], illustration scientifique, **génération procédurale** [W], ***glitch art*** [W]… La barre des 3,5 millions de téléchargements totaux a été dépassée l'an dernier, avec une moyenne actuelle d'environ 400 téléchargements journaliers effectués depuis le site officiel (chiffres en baisse constante depuis quelques années car *G'MIC* est de plus en plus téléchargé et installé via des sources alternatives extérieures).
Il est parfois difficile de garder un rythme soutenu de développement et la motivation qui doit aller avec, mais on s'accroche, en repensant aux utilisateurs heureux qui partagent de temps à autre leur enthousiasme pour ce projet !

On ne peut évidemment pas nommer tous les particuliers, contributeurs à *G'MIC*, que l'on souhaiterait remercier, et avec qui on s'est régalé à échanger durant ces dix années, mais le cœur y est ! Remercions également le laboratoire GREYC et l'institut **INS2I du CNRS** qui affichent un fort soutien à ce projet libre. Un grand merci également

à l'équipe de *LinuxFr.org* qui n'a pas rechigné à relire et publier nos propositions régulières de **dépêches sur *G'MIC*** ;).

Mais cessons de ressasser de vieux souvenirs et passons maintenant aux choses sérieuses : les nouveautés apparues depuis la dernière version majeure 2.2 !

2. Illumination automatique de dessins colorisés en aplats

G'MIC s'est récemment doté d'un filtre assez étonnant, nommé « *Illuminate 2D shape* », dont l'objectif est d'ajouter automatiquement des zones d'illumination et des ombres propres sur des dessins 2D colorisés **en aplats** [W], pour leur donner un aspect 3D.

Dans un premier temps, l'utilisateur fournit un objet à illuminer, sous la forme d'une image sur fond transparent (typiquement un dessin de personnage, ou d'animal). En analysant la forme et le contenu de l'image, *G'MIC* tente alors d'en déduire une carte d'élévations 3D concordante (« *bumpmap* »). La carte d'élévations obtenue est évidemment non exacte, puisqu'un dessin 2D colorisé en aplats ne contient pas d'informations franchement explicites sur sa structure 3D associée ! À partir des élévations 3D estimées, il est aisé d'en déduire une carte de normales (« *normalmap* ») qui est utilisée dans un second temps pour générer un calque d'illumination associé au dessin (suivant un **modèle d'ombrage de Phong** [W]).

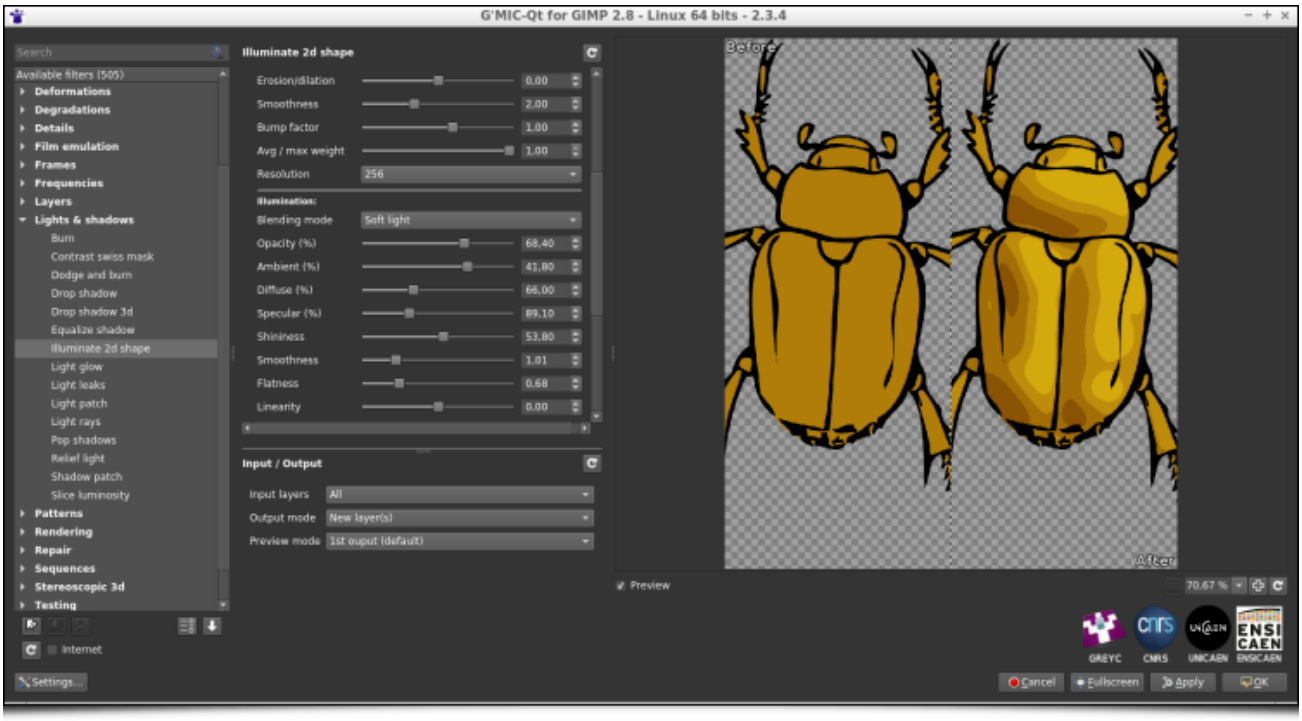


Fig. 2.1 : Le filtre « Illuminate 2D shape » de G'MIC en action, pour ombrer un dessin de scarabée (résultat d'ombrage à droite).

Ce nouveau filtre est très flexible et permet à l'utilisateur d'avoir un contrôle assez fin sur les paramètres d'éclairage (position et type de rendu de la source lumineuse), et d'estimation des élévations 3D. Par ailleurs, le filtre laisse à l'artiste le loisir de retravailler le calque d'illumination généré, ou même directement les cartes d'élévations et de normales 3D estimées. La figure ci-dessous illustre le processus dans son ensemble : à partir de l'image de scarabée colorisé en aplats (*en haut à gauche*), le filtre estime de manière complètement automatique une carte de normales 3D associée (*en haut à droite*), ce qui lui permet de générer des rendus d'illumination du dessin (*ligne du bas*, avec deux styles de rendus différents : lisse et quantifié).

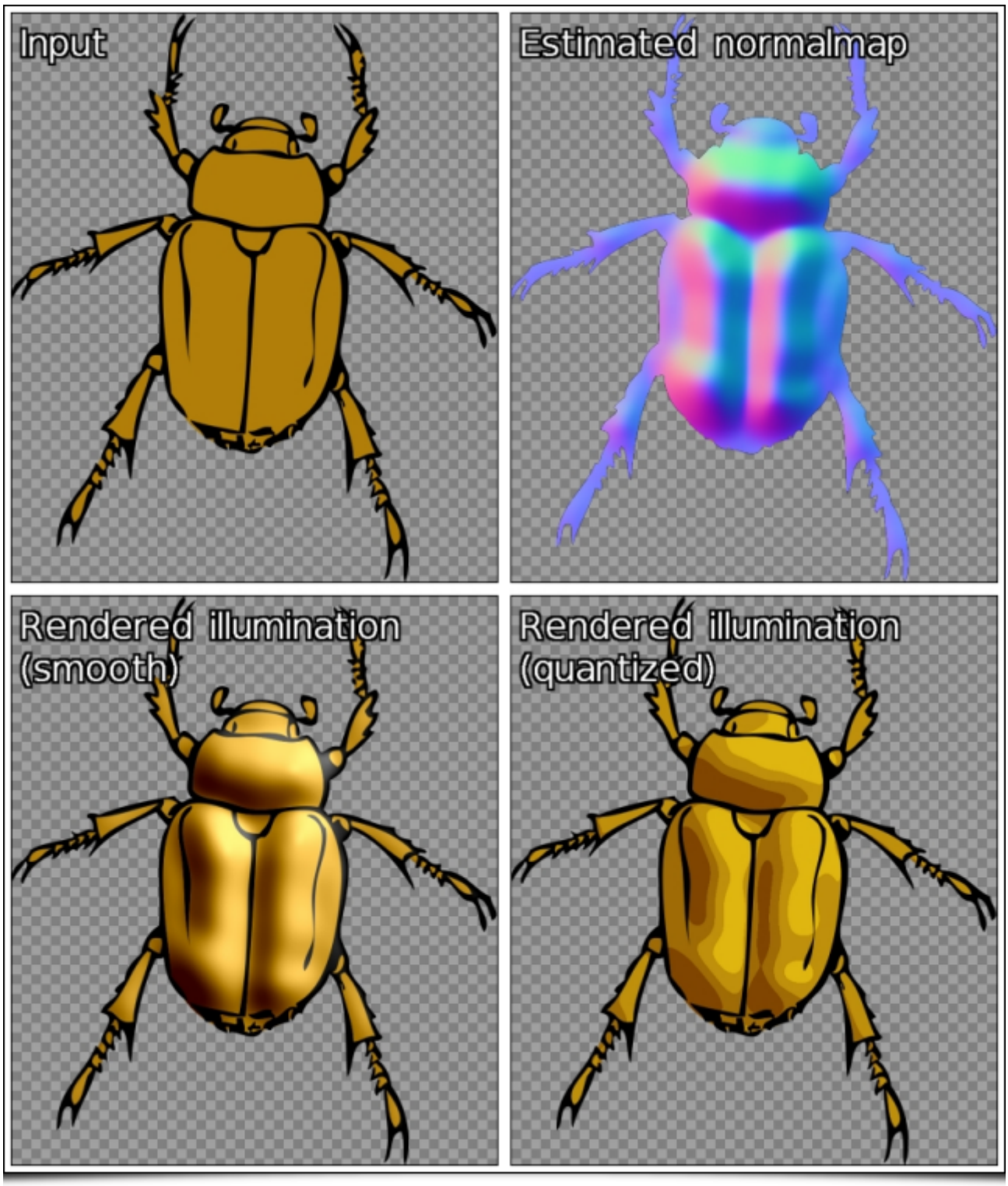


Fig. 2.2 : Le fonctionnement du filtre « Illuminate 2D shape » de G'MIC passe par l'estimation d'une carte de normales 3D pour générer l'illumination automatique d'un dessin.

Malgré la difficulté inhérente au problème de conversion d'une image 2D en informations d'élévations 3D, l'algorithme utilisé s'avère étonnamment efficace dans pas mal de cas, l'estimation de la carte d'élévations 3D obtenue étant suffisamment cohérente pour générer automatiquement des illuminations de dessins 2D plausibles, comme illustré par les deux exemples ci-dessous, obtenus en quelques clics seulement !

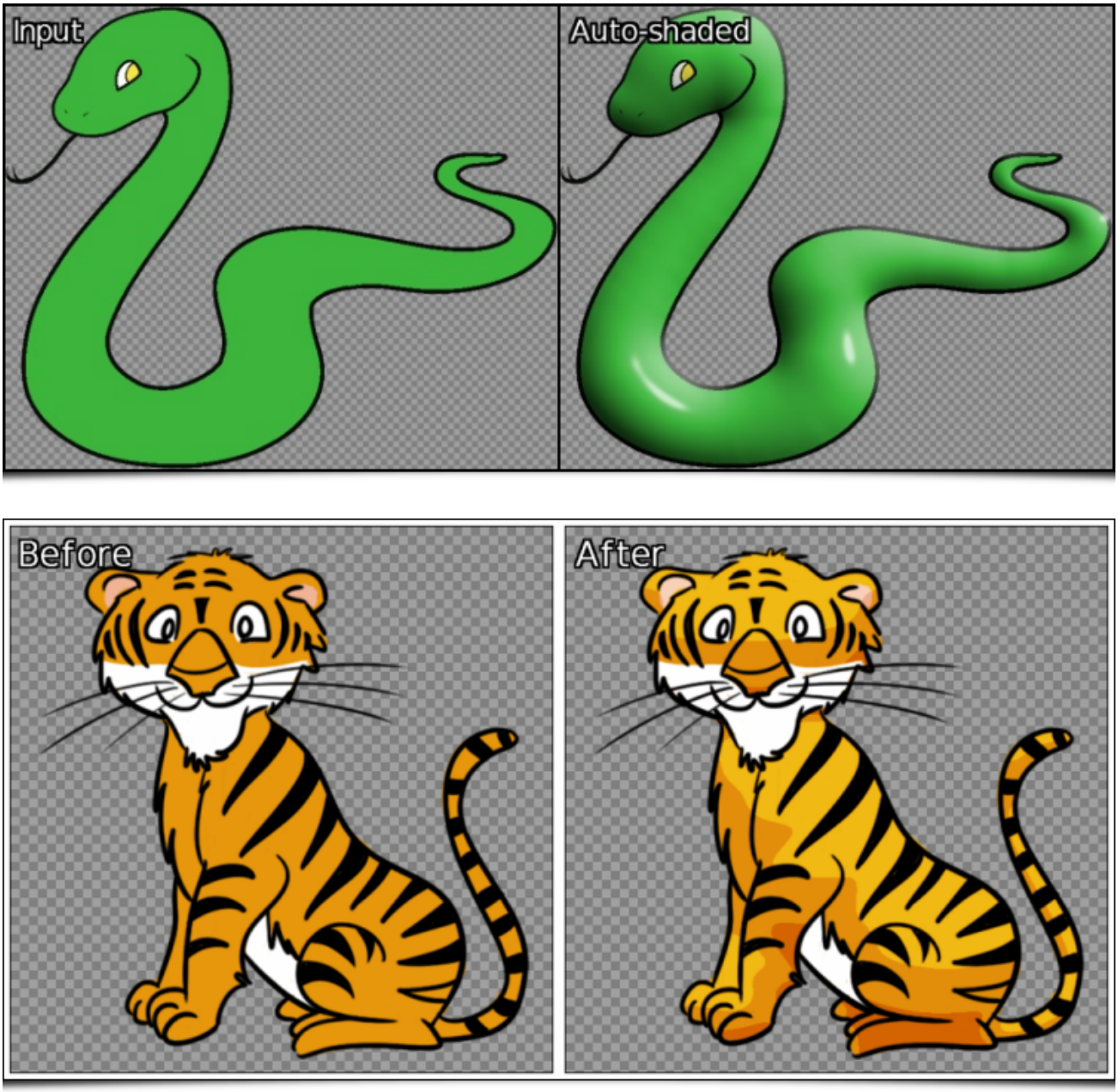


Fig. 2.3 : Deux exemples d'illuminations complètement automatiques de dessins 2D, générés par G'MIC.

Il arrive, bien sûr, que la carte d'élévations 3D estimée ne corresponde pas tout à fait à ce que l'on pourrait souhaiter. Qu'à cela ne tienne, le filtre permet à l'utilisateur de fournir des « guides », sous la forme d'un calque additionnel composé de traits colorés, donnant des informations plus précises à l'algorithme sur la structure du dessin à analyser. La figure ci-dessous illustre l'utilité de ces guides pour un exemple d'illumination d'un dessin d'une main (*en haut à gauche*) : l'illumination obtenue de manière complètement automatique (*en haut à droite*) ne prend pas en compte les informations de lignes de la main. Inclure ces quelques lignes dans un calque additionnel de « guides » (en rouge, *en bas à gauche*) permet d'aider l'algorithme à illuminer le dessin de manière plus satisfaisante.

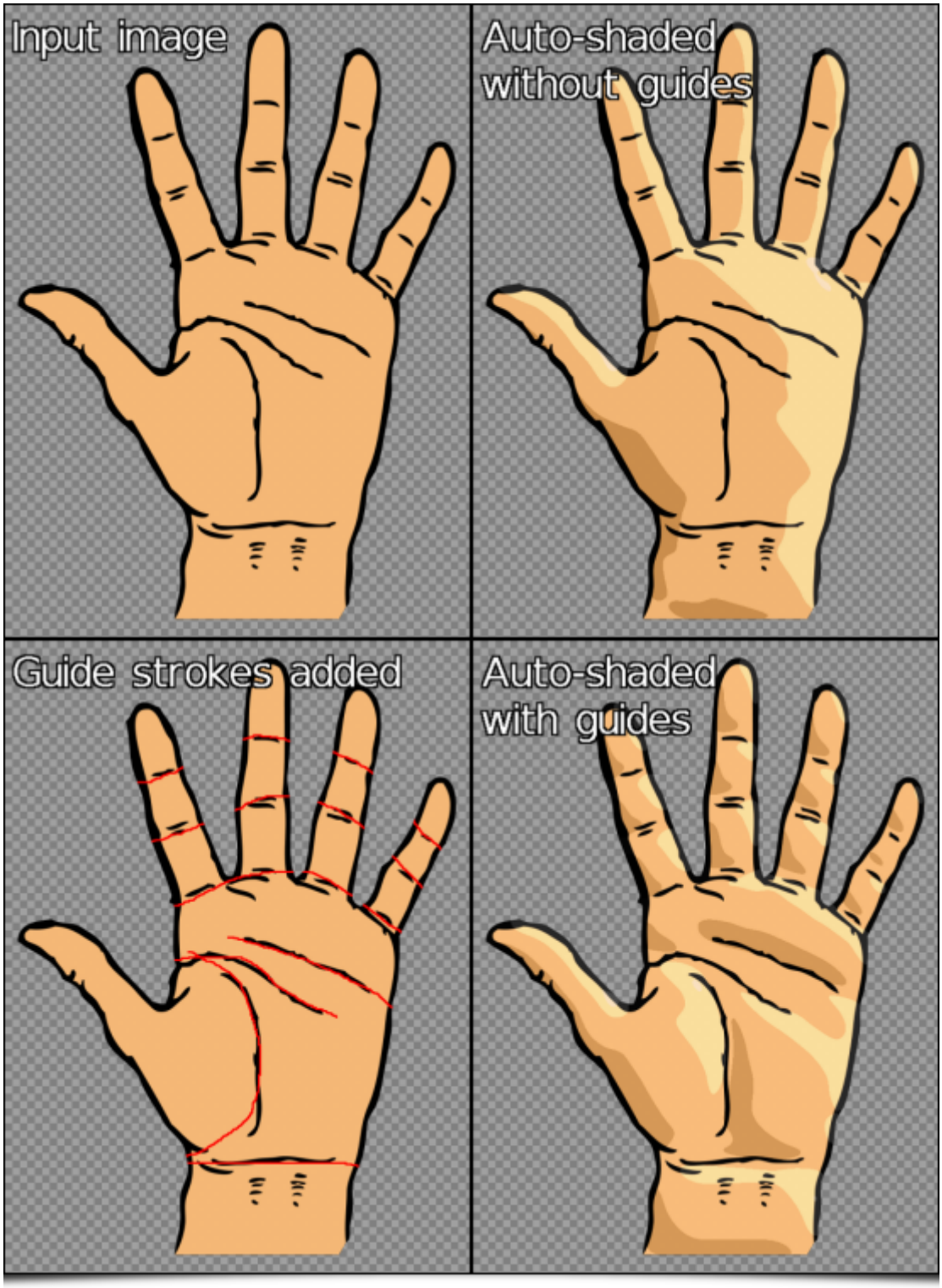


Fig. 2.4 : Utilisation d'un calque de « guides » pour améliorer le rendu d'illumination automatique généré par G'MIC.

Si l'on analyse plus précisément les différences obtenues entre les cartes d'élévations 3D estimées avec et sans « guides » (illustrées ci-dessous sous forme d'objets 3D symétrisés), il n'y a pas photo : on passe d'une grosse moufle boudinée à une estimation 3D de main nettement plus détaillée !

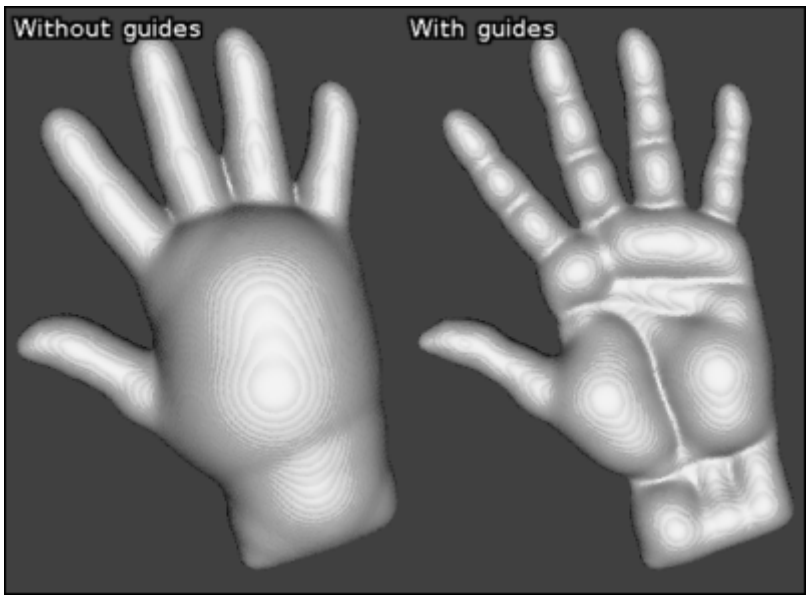


Fig. 2.5 : Élévations 3D estimées pour le dessin précédent de la main, avec et sans utilisation de « guides ».

Notons pour finir que ce filtre dispose également d'un mode de prévisualisation **interactif**, permettant à l'utilisateur de faire bouger la source lumineuse (à la souris) et d'avoir un aperçu du dessin illuminé en temps réel. En modifiant les paramètres de position de la source lumineuse, il est ainsi possible d'obtenir le type d'animations ci-dessous en très peu de temps, qui donne une idée assez précise de la structure 3D estimée par l'algorithme à partir du dessin d'origine.

Fig. 2.6 : Modification de la position de la source lumineuse et rendus d'illumination associés, calculés de manière automatique par G'MIC.

Une vidéo montrant les différentes possibilités d'édition de l'illumination permises par ce filtre est **visible sur YouTube**. Espérons que cette nouvelle fonctionnalité de *G'MIC* permette aux artistes d'accélérer l'étape d'illumination et d'ombrage de leurs futurs dessins !

3. Projection stéréographique

Dans un tout autre genre, nous avons également ajouté à *G'MIC* un filtre implémentant la **projection stéréographique** [W], très précisément nommé « *Stereographic projection* ». Ce type de projection cartographique permet de projeter des données images définies sur une sphère, sur un plan. Il faut savoir que c'est la projection

usuelle utilisée pour générer des images de « mini-planètes » à partir de panoramas équirectangulaires, comme celui illustré sur la figure ci-dessous.

*Fig. 3.1 : Exemple de panorama équirectangulaire (réalisé par **Alexandre Duret-Lutz**).*

Si, sur ce panorama, on lance le greffon *G'MIC* et que l'on sélectionne le filtre « *Stereographic projection* », on obtient :

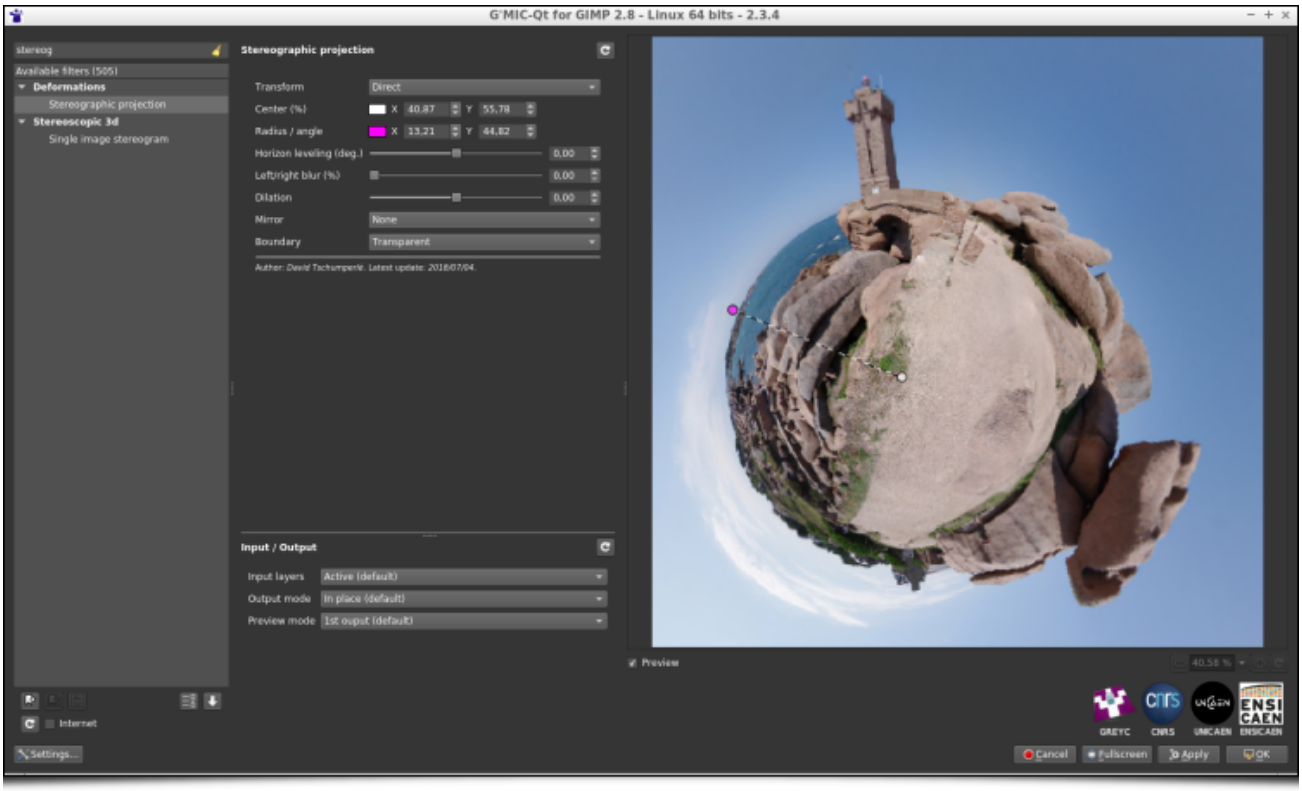


Fig. 3.2 : Le filtre « Stereographic projection » de G'MIC en action dans le greffon pour GIMP ou Krita.

Le filtre permet des réglages précis du centre de projection, de l'angle de rotation et du rayon de la sphère considérée, tout ça de manière interactive directement sur la fenêtre de prévisualisation (nous y reviendrons par la suite). En quelques clics, et après application du filtre, nous obtenons la « mini-planète » désirée :

Fig. 3.3 : « Mini-planète » obtenue après projection stéréographique.

Il est d'ailleurs cocasse de constater qu'en inversant simplement l'axe vertical des images, on transforme une « mini-planète » en un « maxi-tunnel » !

Fig. 3.4 : « Maxi-tunnel » obtenu par inversion de l'axe vertical puis projection stéréographique.

Là encore, nous avons réalisé **cette petite vidéo** qui montre ce filtre en conditions réelles d'utilisation. À noter que *G'MIC* possédait déjà un filtre similaire (dénommé « *Sphere* »), qui pouvait être utilisé pour la création de « mini-planètes », mais avec un type de projection moins bien adapté que la projection stéréographique, qu'il est maintenant possible d'utiliser.

4. Toujours plus de possibilités pour la manipulation des couleurs

Triturer les couleurs des images est une occupation récurrente chez les photographes et les illustrateurs, et *G'MIC* possèdait déjà plusieurs dizaines de filtres destinés à cette unique activité, regroupés dans une catégorie dédiée (à savoir « *Colors/* », pour faire original !). Cette catégorie s'étoffe encore, avec deux nouveaux filtres fraîchement apparus.

4.1. Le filtre « CLUT from after-before layers »

Le filtre « **CLUT from after-before layers_ » cherche à modéliser la transformation colorimétrique qui a été effectuée entre deux images (que l'on possède). Supposons par exemple que nous ayons la paire d'images suivante :

Fig. 4.1 : Paire d'images où une transformation colorimétrique inconnue a été appliquée sur l'image du haut, pour obtenir celle du bas.

Problème : On ne se rappelle plus du tout comment on a fait pour passer de l'image originale à l'image modifiée, mais on voudrait absolument ré-appliquer le même processus sur une autre image. Eh bien, plus de soucis, appelons *G'MIC* à la rescousse ! Le filtre en question va chercher à modéliser au mieux la modification des couleurs sous la forme d'une ***HaldCLUT***, qui se trouve être une façon classique de représenter une transformation colorimétrique quelconque.

Fig. 4.2 : Le filtre modélise la transformation colorimétrique entre deux images sous forme d'une HaldCLUT.

La *HaldCLUT* générée par le filtre va pouvoir être sauvée et ré-appliquée sur d'autres images, avec la propriété désirée que l'application de cette *HaldCLUT* sur l'image originale redonne bien l'image modèle cible dont on s'est servi pour que le filtre apprenne la transformation couleur produite.
À partir de là, nous sommes capables d'appliquer une modification de couleurs équivalente, sur n'importe quelle autre image :

Fig. 4.3 : La transformation colorimétrique estimée sous forme de HaldCLUT est ré-appliquée sur une autre image.

Ce filtre permet donc *in fine* de créer des *HaldCLUT* « par l'exemple », et pourrait donc intéresser de nombreux photographes (notamment ceux qui diffusent des compilations de fichiers *HaldCLUT*, **librement** ou non !).

4.2. Filtre « Mixer [PCA] »

Un deuxième filtre de manipulation de couleurs, nommé « *Mixer [PCA]* » a été aussi récemment intégré à *G'MIC*. Il agit comme un classique **mixeur de canaux couleurs**, mais plutôt que de travailler dans un espace couleur prédéfini (comme sRGB, HSV, Lab…), il agit sur l'espace couleur « naturel » de l'image d'entrée, obtenue par **analyse en composante principale**[W] (*ACP*) de ses couleurs RVB. Ainsi à chaque image sera associé un espace couleur différent.
Par exemple, si nous prenons l'image « lion » ci-dessous, et que l'on regarde la distribution de ses couleurs dans le cube RVB (image de droite), on s'aperçoit que l'axe principal de variation des couleurs est défini par une droite allant du orange foncé au beige clair (axe symbolisé par la flèche rouge sur la figure).

Fig. 4.4 : Distribution des couleurs de l'image « lion » dans le cube RVB, et axes principaux associés (colorisés en rouge, vert et bleu).

L'axe de variation secondaire quant à lui (flèche verte) va du bleu jusqu'à l'orange, et l'axe tertiaire (flèche bleue) du vert au rose. Ce sont ces axes de variations (plutôt que les axes RVB) qui vont donc définir la base de couleurs utilisée dans ce filtre de mixage de canaux.

Fig. 4.5 : Le filtre « Mixer [PCA] » est un mixeur de canaux agissant sur les axes de variations de couleurs « naturels » de l'image.

Il serait malhonnête d'affirmer qu'il soit toujours meilleur de considérer la base couleur obtenue par ACP pour le mixage des canaux, et ce nouveau filtre n'a évidemment pas vocation à être le mixeur « ultime » qui remplacerait tous les autres. Il a simplement le mérite d'exister et de proposer une alternative aux outils usuels de mixage de canaux couleur, alternative dont les résultats se sont avérés effectivement intéressants sur plusieurs images de tests utilisées lors du développement de ce filtre. Cela ne coûte rien d'essayer en tout cas…

5. Méli-mélo de filtres

Cette section présente pêle-mêle quelques autres nouveaux filtres et améliorations diverses qui ont été intégrés récemment à *G'MIC* et qui méritent qu'on les mentionne, sans s'y attarder outre mesure.

5.1. Le filtre « Local processing »

Ce filtre permet d'appliquer un processus de normalisation ou d'égalisation de couleurs sur des voisinages locaux d'image (avec éventuellement du recouvrement entre voisinages). C'est un filtre supplémentaire permettant de faire ressortir des détails dans des photographies initialement surexposées ou sous-exposées, mais qui peut parfois créer des « halos » disgracieux.

Fig. 5.1 : Le filtre « Local processing » permet de rehausser les détails dans des photographies sur ou sous-exposées.

5.2. Le filtre « Blend [standard] »

Vous trouvez que vous n'avez pas assez de modes de fusion de calques à votre disposition dans GIMP ou Krita ? Vous rêvez de pouvoir définir votre propre formule de fusion ? Alors le filtre « *Blend [standard]* » est fait pour vous ! Ce filtre, déjà existant auparavant, s'enrichit de la fonctionnalité « *Custom formula* » qui permet à l'utilisateur de spécifier sa propre **formule mathématique** de fusion de calques. Toutes les fantaisies deviennent possibles !

Fig. 5.2 : Le filtre « Blend [standard] » permet maintenant de définir ses propres formules mathématiques de fusion de calques.

5.3. Le filtre « Sketch »

Signalons aussi la ré-implémentation complète du sympathique filtre « *Sketch* », qui existait depuis plusieurs années, mais qui pouvait s'avérer un peu lent sur de grosses images. La nouvelle implémentation est beaucoup plus rapide, tirant notamment parti du calcul multicœur quand c'est possible.

Fig. 5.3 : Le filtre « Sketch » a été réimplémenté et exploite maintenant tous les cœurs de calcul disponibles.

5.4. Le filtre « Mandelbrot - Julia sets »

Un gros travail de ré-implémentation a été également réalisé sur le filtre « *Mandelbrot - Julia sets* », puisque l'interface de navigation a été entièrement repensée, rendant l'exploration de l'**ensemble de Mandelbrot** [W] bien plus confortable (comme l'illustre cette**vidéo**). De nouvelle options pour le choix des couleurs sont également apparues.

Fig. 5.4 : Le filtre « Mandelbrot - Julia sets » et sa nouvelle interface de navigation dans l'espace complexe.

5.5. Le filtre « Polygonize [Delaunay] »

Le filtre « *Polygonize [Delaunay]* » qui génère des rendus polygonisées d'images couleurs se dote d'un nouveau mode de rendu, utilisant des couleurs interpolées linéairement dans les **triangles de Delaunay** [W] produits.

Fig. 5.5 : Les différents modes de rendu du filtre « Polygonize [Delaunay] ».

6. Autres faits marquants du projet

6.1. Améliorations de l'interface du greffon

Bien sûr, les nouveautés dans *G'MIC* ne concernent pas seulement les filtres de traitement d'images proprement dits ! Un travail considérable a été par exemple réalisé sur l'interface graphique du greffon *G'MIC-Qt*.

Les filtres du greffon ont désormais la possibilité de spécifier un nouveau type de paramètre `point()`, qui se matérialise sous la forme d'un petit disque coloré que l'on peut manipuler directement à la souris au-dessus de la fenêtre de prévisualisation. En pratique, cela permet de rendre cette fenêtre de prévisualisation interactive, et ce n'est pas rien ! De nombreux filtres utilisent maintenant cette capacité, ce qui les rend beaucoup plus agréables et intuitifs à utiliser (voir **cette vidéo** pour quelques exemples). L'animation ci-dessous montre par exemple comment ces points interactifs sont utilisés dans le nouveau filtre « *Stereographic projection* », que nous avons décrit précédemment.

Fig. 6.1 : La fenêtre de prévisualisation du greffon G'MIC-Qt s'enrichit de nouvelles possibilités d'interactions pour l'utilisateur.

L'introduction de cette fonctionnalité a de ce fait permis d'améliorer les modes de division de prévisualisation (« *split preview* »), utilisés par un grand nombre de filtres pour afficher côte à côte les images « avant / après » lors de la prévisualisation d'un filtre dans le greffon. Il est maintenant possible de déplacer la zone frontière des images « avant / après », comme illustré par l'animation ci-dessous. Deux nouveaux modes de division, en damier, ont d'ailleurs été ajoutés à cette occasion.

Fig. 6.2 : Les modes de division de la prévisualisation possèdent maintenant une frontière « avant / après » déplaçable.

Plein d'autres petites améliorations ont été faites dans le code du greffon : une prise en charge du dernier GIMP 2.10, de la version Qt 5.11, une meilleure gestion des messages d'erreurs s'affichant dans la fenêtre de prévisualisation, un *design* général plus épuré, et tout un tas de petites choses par forcément visibles mais participant néanmoins au confort de l'utilisateur (un système de cache d'images pour la fenêtre de prévisualisation par exemple). Bref, que du bon !

6.2. Perfectionnement du cœur du logiciel

De nouvelles améliorations ont également été apportées dans les couches internes de *G'MIC*.

La « bibliothèque standard » du langage de script *G'MIC* évolue, avec l'apparition de nouvelles commandes pour le calcul des fonctions hyperboliques inverses (`acosh`, `asinh` et `atanh`), ainsi que de la commande `tsp` (***t**ravelling **s**alesman **p**roblem*) qui estime une solution « acceptable » au problème bien connu du **voyageur de commerce**[W], et ceci pour un nuage de points de dimension et de taille quelconque.

Fig. 6.3 : Estimation du circuit le plus court entre plusieurs centaines de points 2D, par la commande `tsp` de G'MIC.

Fig. 6.4 : Estimation du circuit le plus court entre plusieurs couleurs du cube RVB (en 3D donc), grâce à la commande `tsp` de G'MIC.

L'interface de démonstration, qui se lance lorsque l'on invoque `gmic` sans arguments à partir de la ligne de commande, a également été refaite en repartant de zéro.

Fig. 6.5 : La nouvelle interface de démonstration de `gmic`, l'outil en ligne de commande de G'MIC.

Le compilateur *JIT* d'expressions mathématiques intégré n'est pas non plus en reste et s'enrichit de nouvelles fonctions permettant de tracer des polygones (fonction `polygon()`) ou des ellipses (fonction `ellipse()`) dans des images. Ces expressions mathématiques peuvent en réalité définir de véritables petits programmes (possédant des variables locales, des fonctions utilisateur et des structures de contrôle). On peut ainsi générer des images synthétiques facilement, avec de simples lignes de commande, comme le montrent les deux exemples ci-dessous.

```
$ gmic 400,400,1,3 eval "for (k = 0, k<300, ++k, polygon(3,u([vector10(0),[w,h,w,h,w,h,0.5,255,255,255
```

Résultat :

Fig. 6.6 : Utilisation de la nouvelle fonction `polygon()` du compilateur à la volée de G'MIC, pour générer une image synthétique aléatoire.

```
$ gmic 400,400,1,3 eval "for (k=0, k<20, ++k, ellipse(w/2,h/2,w/2,w/8,k*360/20,0.1,255))"
```

Résultat :

Fig. 6.7 : Utilisation de la nouvelle fonction `ellipse()` du compilateur à la volée de G'MIC, pour générer une image synthétique florale.

Notons également une meilleure gestion des valeurs `NaN`[W] lors des calculs réalisés par le cœur du logiciel, ce qui permet à *G'MIC* d'avoir un comportement cohérent même lorsqu'on le compile avec l'optimisation `-ffast-math`. Ainsi, *G'MIC* est maintenant compilable sans soucis avec le niveau d'optimisation maximal `-Ofast` du compilateur *g++*, alors que nous étions restreints dans le passé à utiliser « seulement » `-O3`. L'amélioration en vitesse de calcul se fait clairement ressentir pour certains filtres !

6.3. Supports de diffusion

Pas mal de changements ont aussi eu lieu sur les supports de diffusion utilisés par le projet.

Tout d'abord, les pages Web du projet (qui en passant utilisent maintenant des connexions sécurisées HTTPS par défaut) s'enrichissent d'une nouvelle **galerie d'images**. Cette galerie montre à la fois des résultats d'application de différents opérateurs de traitement d'images disponibles dans *G'MIC*, mais aussi la façon de les reproduire (à partir de la ligne de commande). Notons d'ailleurs que ces pages de galerie sont générées automatiquement par un script *G'MIC* dédié à cette tâche, ce qui nous assure que la syntaxe donnée pour chaque exemple est exacte.

Fig. 6.8 : La nouvelle page de galerie d'images du site Web de G'MIC.

Cette galerie est découpée en plusieurs sections, suivant le type de traitements effectué (*Artistique, Noir & blanc, Déformation, Filtrage, etc.*). La dernière section « ***Code sample*** » est personnellement celle que je trouve la plus

amusante, puisqu'elle présente de petites séquences d'images (sous forme de GIF animés qui bouclent) entièrement générées par des scripts courts en langage *G'MIC*. Une façon un tout petit peu exotique d'utiliser *G'MIC*, mais qui montre son potentiel pour l'**art génératif**[W].

Exemple 1 Exemple 2

Fig. 6.9 : Deux exemples d'animations GIF générées par des scripts en langage G'MIC, visibles dans la galerie d'images.

Et puis, nous avons déménagé le dépôt source *git* principal du projet vers **Framagit**, en gardant néanmoins un miroir synchronisé sur GitHub au même emplacement qu'auparavant (pour profiter en particulier du fait que de nombreux développeurs sont présents sur GitHub et peuvent plus facilement créer un dépôt divergent (*fork*) et nous faire des rapports de bogues sur cette plate-forme).

7. Conclusions et perspectives

Voilà ! Notre tour des nouveautés (des six derniers mois d'activité) du projet *G'MIC* s'achève enfin.

Nous sommes heureux d'avoir pu vivre dix belles années d'émotions informatiques avec la naissance et l'évolution de ce projet libre, et de pouvoir partager à notre manière, avec tous les utilisateurs, des techniques de traitements d'images avancées. On espère surtout repartir de plus belle pour de nombreuses années ! Les soutiens se font de plus en plus présents autour de nous, donc on se dit que ça doit pouvoir se faire (à ce propos, si vous voulez contribuer au projet de quelque manière que ce soit, vous êtes les bienvenus !).
Notons que l'année prochaine, on fêtera également les vingt ans d'existence de ***CImg***, la bibliothèque *C++* de traitement d'images qui est directement à l'origine du projet *G'MIC* (et qui, elle, est née en novembre 1999, ça ne nous rajeunit pas ma bonne dame…). Preuve s'il en est que l'intérêt du logiciel libre, c'est qu'il s'inscrit dans la durée !

Et en attendant la prochaine dépêche sur *G'MIC*, n'hésitez pas à tester ce logiciel, à jouer et triturer vos images de la manière la plus libre et créative possible !

Aller plus loin

- **Le projet G'MIC** (685 clics)
- **Fil Twitter** (106 clics)
- **Annonce de la toute première version de G'MIC sur LinuxFr.org** (166 clics)
- **Série d'articles G'MIC sur LinuxFr.org** (186 clics)

Bon anniversaire et merci
Posté par **Anthony Jaguenaud** le 21/08/18 à 10:35. Évalué à 7.

> Voila,
> Bon anniversaire et merci. Je lisais déjà Linuxfr en 2008, mais je n'ai retenu G'MIC qu'à partir du moment où tu as présenté des images des filtres…
>
> Je me demande s'il est possible d'exporter le model de la main en 3D (facettes) ? pour une réutilisation dans blender par exemple.

Re: Bon anniversaire et merci
Posté par **David Tschumperlé** (**site web personnel**) le 21/08/18 à 13:49. Évalué à 10.