G'MIC 3.2.5 : 15 ans de développement pour du traitement d'images libre et reproductible

Posté par **David Tschumperlé** (**site web personnel**) le 30/05/23 à 21:22. Édité par **5 contributeurs**. Modéré par **Ysabeau**. **Licence CC By-SA.**

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

À l'occasion de la sortie de la version **3.2.5** de ***G'MIC*** (*GREYC's Magic for Image Computing*), **cadriciel** [W] libre pour **le traitement des images** [W], nous vous proposons un récapitulatif des nouvelles fonctionnalités implémentées depuis notre **précédente dépêche** (publiée en décembre 2021). C'est aussi pour nous l'opportunité de célébrer les **15 ans** d'existence du projet !

G'MIC est développé à **Caen** [W], en France, dans l'équipe ***IMAGE*** du ***GREYC***, un laboratoire public de recherche en Sciences et Technologies de l'Information et de la Communication (Unité Mixte de Recherche ***CNRS*** / ***ENSI-CAEN*** / **Université de Caen**). Il est distribué sous licence libre ***CeCILL***.

Dans cette dépêche, nous détaillerons quelques-unes des fonctionnalités récemment ajoutées, et nous les illustrerons par des exemples de traitement et de synthèse d'images 2D et 3D.

N. D. A. : Cliquez sur les images pour en obtenir une version en pleine résolution, ou une vidéo correspondante lorsque les images contiennent l'icône

Sommaire

1. Qu'est-ce que *G'MIC* ?

G'MIC est un cadriciel (*framework*) libre pour la manipulation et le traitement des **images numériques** [W]. Il propose des interfaces utilisateur variées pour la manipulation algorithmique d'images et de signaux. Le cœur de ce projet repose sur l'implémentation d'un langage de script (le ***« langage G'MIC »***), élaboré spécifiquement pour faciliter le prototypage et l'implémentation de nouveaux algorithmes et opérateurs de traitement d'images. Les utilisateurs peuvent appliquer des opérateurs parmi plusieurs centaines déjà implémentées, mais ont également la possibilité d'écrire leurs propres pipelines de traitement et de les rendre accessibles dans les différentes interfaces utilisateur du projet. C'est donc, par essence, un cadriciel ouvert, extensible et en évolution constante.

Les interfaces utilisateurs de *G'MIC* les plus abouties sont : `gmic`, l'interface en ligne de commande (complément utile à ***ImageMagick*** ou ***GraphicsMagick*** pour ceux qui aiment utiliser le terminal), le service Web ***G'MIC Online***, et surtout, le greffon ***G'MIC-Qt***, utilisable dans de nombreux logiciels populaires d'édition d'images numériques tels que ***GIMP***, ***Krita***, ***DigiKam***, ***Paint.net***, ***Adobe Photoshop*** [W], ***Affinity Photo*** [W]… Ce greffon, très facile à utiliser, propose aujourd'hui plus de **580 filtres** de traitement pour enrichir ces logiciels de manipulation d'images.

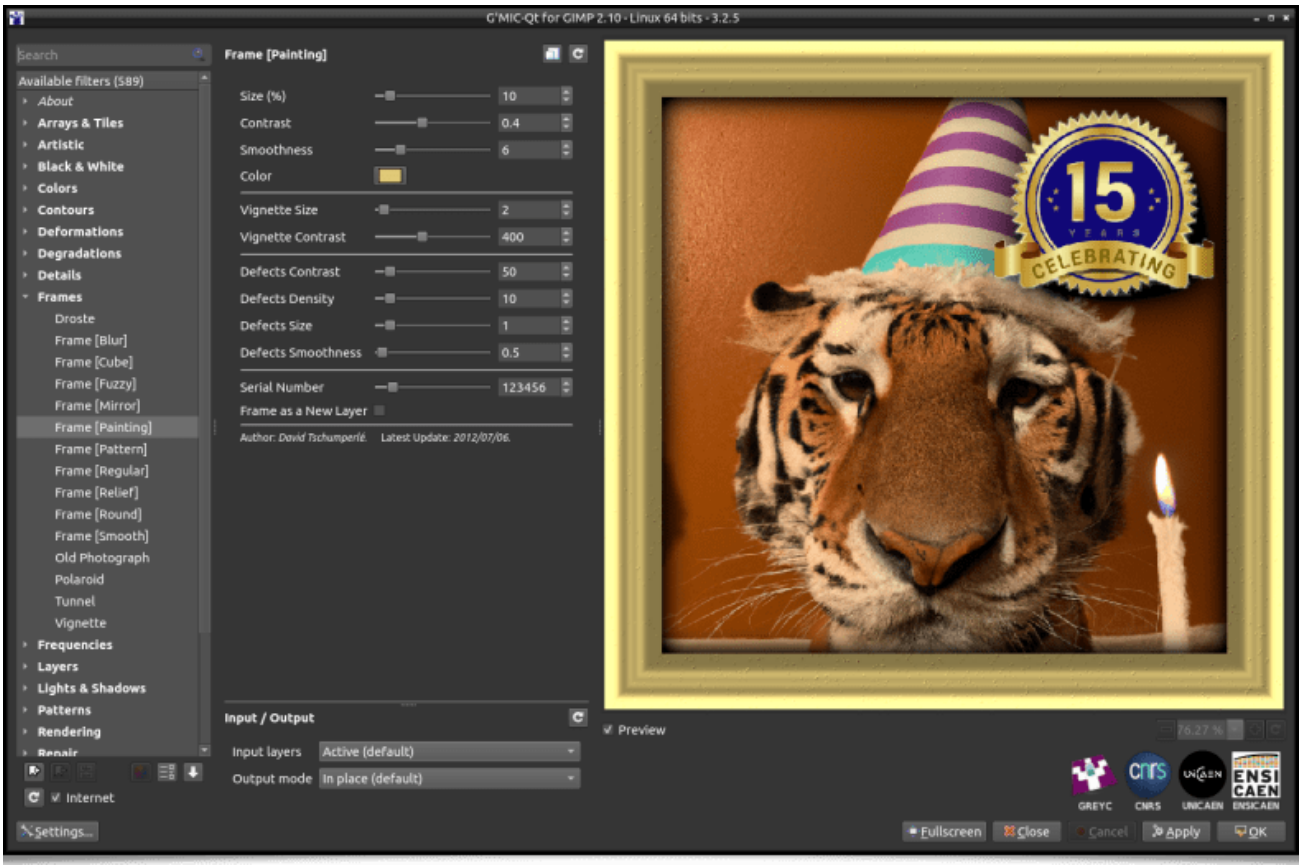

*Fig. 1.1. Aperçu du greffon G'MIC-Qt, en version **3.2.5**, ici lancé depuis GIMP 2.10.*

Grâce à son langage de script dédié, de nouveaux filtres et effets pour le traitement d'images sont régulièrement ajoutés à *G'MIC*.

Dans cette dépêche, nous détaillerons quelques-uns de ces nouveaux traitements et donneront quelques nouvelles du projet. Nous donnerons également des exemples d'utilisation de l'outil en ligne de commande `gmic`, qui est de loin l'interface la plus puissante offerte par le projet.

2. Nouveaux filtres pour l'abstraction, le *Glitch Art* et la génération de motifs

- Pour commencer cette revue des nouveautés, mentionnons l'existence d'un nouveau filtre de transformation d'images sous la forme de dessins au trait (« **Line Art** [W] » en anglais). Ce filtre, opportunément nommé

Artistic / Line Art a été élaboré par **Claude Lion**, contributeur extérieur déjà auteur de plusieurs filtres (dont le très apprécié **Artistic / Comic Book** dont nous **avions déjà parlé** dans notre dépêche précédente).

Ce filtre analyse la géométrie des structures principales dans les images et décide si ces structures doivent apparaître dans une image redessinée sur fond blanc, soit sous forme de traits noirs, soit sous forme d'aplats gris ou noirs. Il fonctionne particulièrement bien avec des portraits, puisque les contrastes sont assez marqués dans ce type d'images.

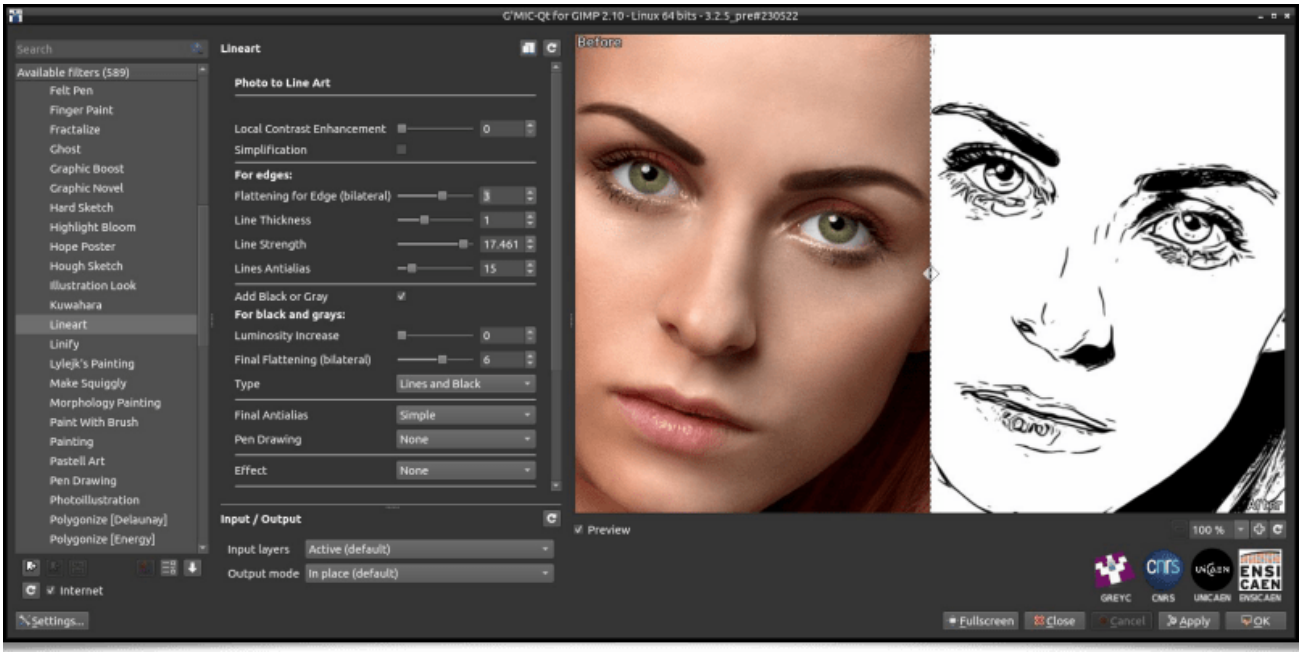

*Fig. 2.1. Le filtre **Artistic / Line Art**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

La visualisation interactive du greffon *G'MIC-Qt* facilite le réglage de l'ensemble des paramètres du filtre, pour personnaliser le type de rendu souhaité. L'appui sur les boutons « *Apply* » ou « *OK* » applique le filtre sur l'image. Notons qu'une fois ces paramètres choisis, l'appui sur le bouton *« Copy to Clipboard »* de l'interface du greffon va copier la commande *G'MIC* correspondante dans le presse-papier.

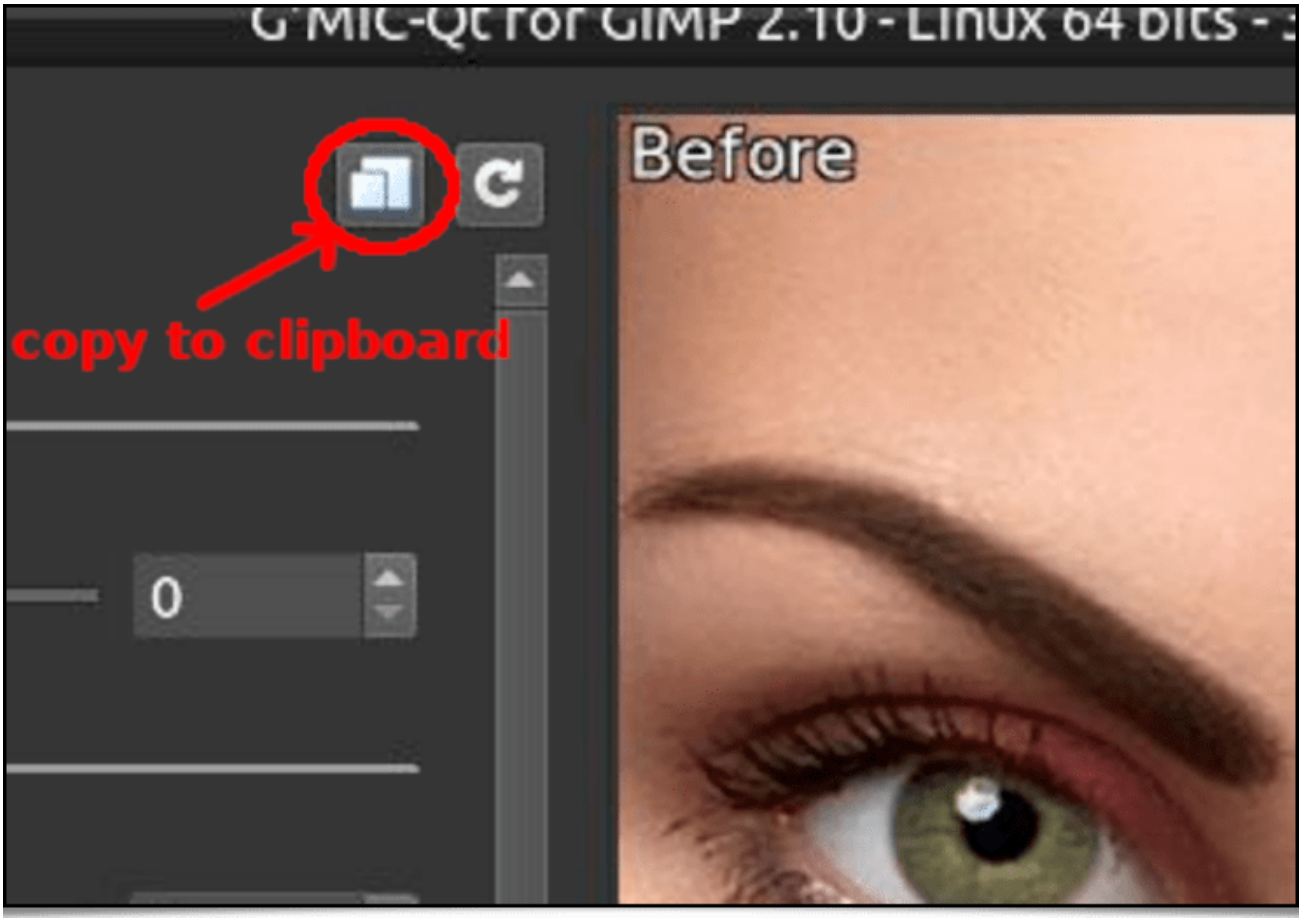

Fig. 2.2. Le bouton « Copy to Clipboard » copie dans le presse-papier la commande G'MIC correspondant à l'action du filtre.

Pour appliquer ensuite le filtre avec les mêmes paramètres sur d'autres images (par exemple pour du traitement par lot), il suffit de lancer `gmic` dans son terminal, en y ajoutant le nom du fichier image à traiter et la commande copiée préalablement dans le presse-papier, ce qui donnera par exemple :

```
$ gmic autre_portrait.jpg cl_lineart 0,0,2,1,15,15,1,0,6,2,2,0,0,0,50,50 output lineart.png
```

Cette astuce est utile quand on souhaite utiliser certains effets *G'MIC* simplement dans des scripts personnalisés (cela fonctionne évidemment avec tous les filtres disponibles dans le greffon).

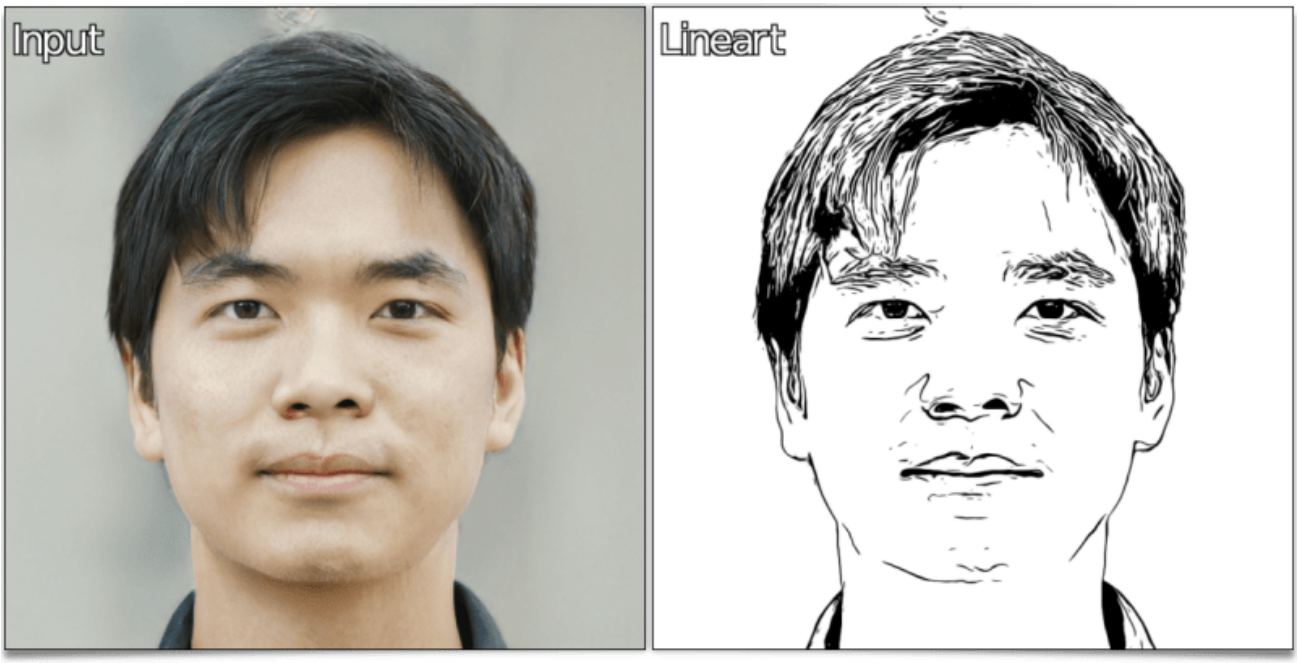

Fig. 2.3. Le filtre « Line Art », appliqué sur une autre image de portrait, avec les mêmes paramètres, depuis le terminal.

Fig. 2.4. Le filtre « Line Art », appliqué sur d'autres images d'exemples.

- Passons maintenant au filtre **Degradations / Huffman Glitches**, un moyen amusant de générer du ***Glitch Art***[W]. Plus précisément, on va simuler ici des artéfacts de décompression d'images via l'ajout volontaire d'erreurs (inversions de bits) dans les **codes d'Huffman**[W] qui auraient été utilisés pour la compression sans perte des données de l'image d'entrée. Cela produit des distorsions numériques visibles sur l'image lors de la décompression des données bruitées, distorsions qui sont justement des effets recherchés par les amateurs de *Glitch Art* !

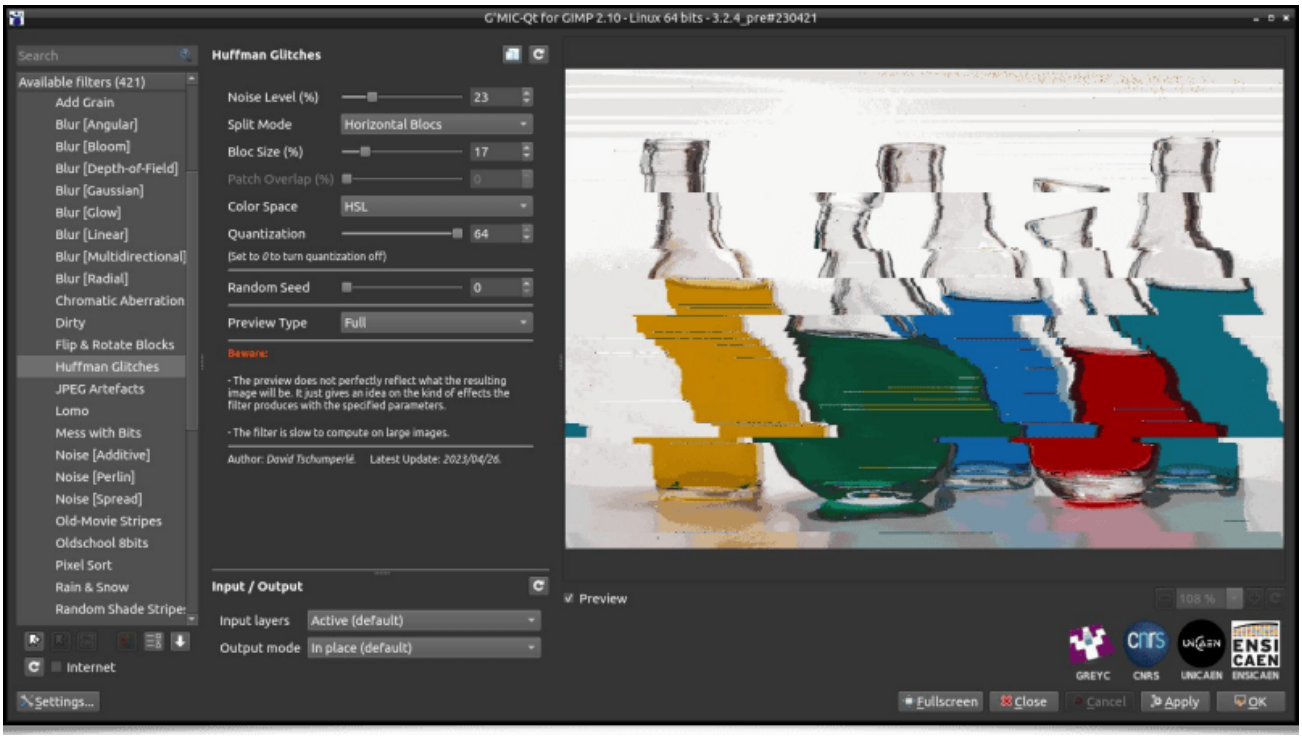

*Fig. 2.5. Le filtre **Degradations / Huffman Glitches**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

Ce filtre permet la génération d'artéfacts de compression avec des variations : bloc par bloc, ligne par ligne, colonne par colonne, ou sur des données image encodées dans des espaces couleur différents de *RGB*. Au final, la variété des anomalies qu'il est possible de produire est assez importante, comme illustré par la figure suivante :

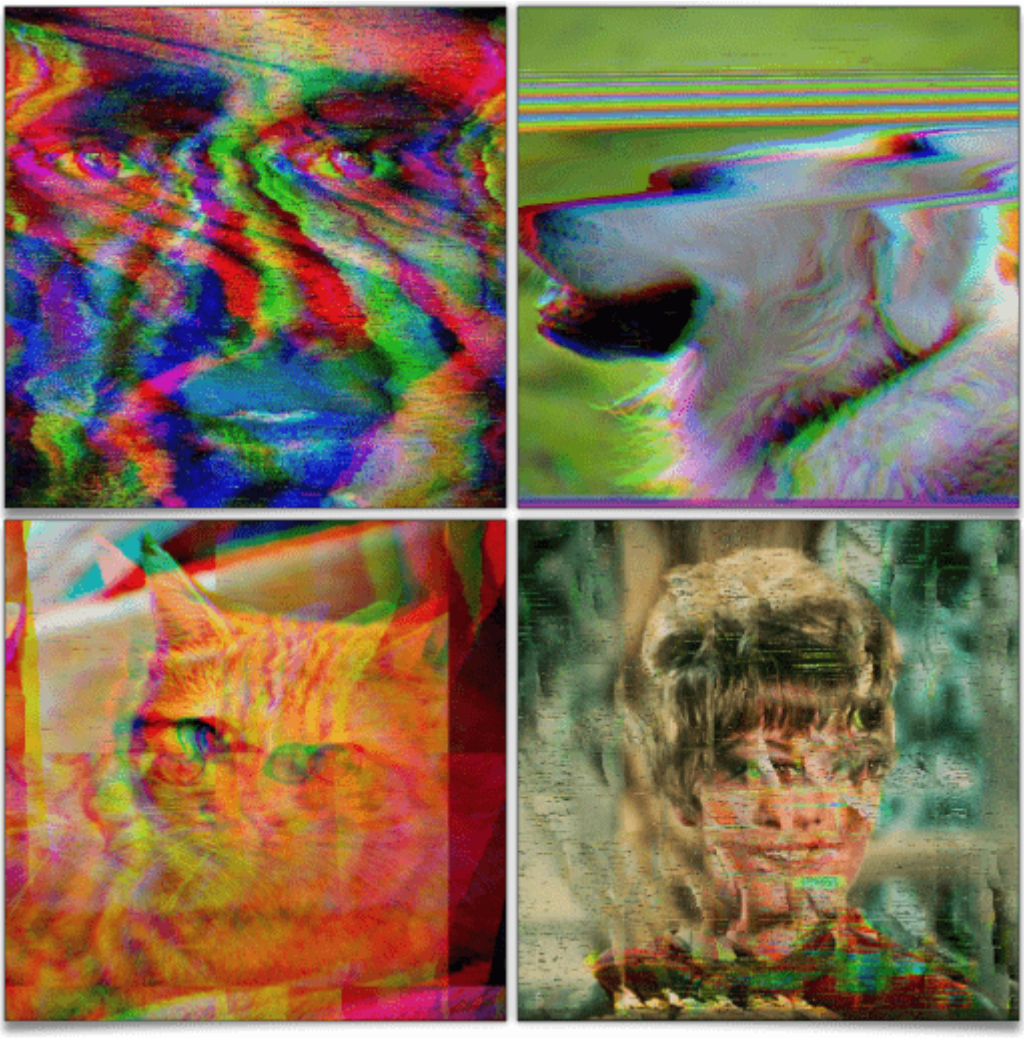

*Fig. 2.6. Quelques variations des paramètres du filtre **Degradations / Huffman Glitches**.*

Là encore, il est facile de récupérer la commande *G'MIC* correspondant à l'application du filtre, pour l'utiliser dans un script, par exemple pour l'application de cet effet sur toutes les *frames* d'une vidéo (cliquez sur l'image ci-dessous pour visualiser la vidéo) :

*Fig. 2.7. Le filtre **Degradations / Huffman Glitches** appliqué sur la vidéo **Tears of Steel** [W] de la fondation Blender.*

Il flotte comme un doux parfum de télé analogique… ☺

- Mentionnons également l'apparition d'un nouveau filtre, nommé **Patterns / Pack Ellipses**, qui risque de ne pas plaire à nos lecteurs **trypophobes** [W] (aucun lien avec la peur de manger des tripes à la mode de Caen) ! Ce filtre a pour tâche de redessiner une image en emboîtant des ellipses colorées, sans les faire se toucher. Les ellipses sont orientées parallèlement ou orthogonalement aux structures locales, pour faire ressortir au mieux les contours les plus saillants des images. Ce n'est pas le premier filtre de ce type dans

G'MIC, mais on a ici un nouvel algorithme d'**empilement compact** [W], relativement rapide à exécuter, et qui produit des images intéressantes.

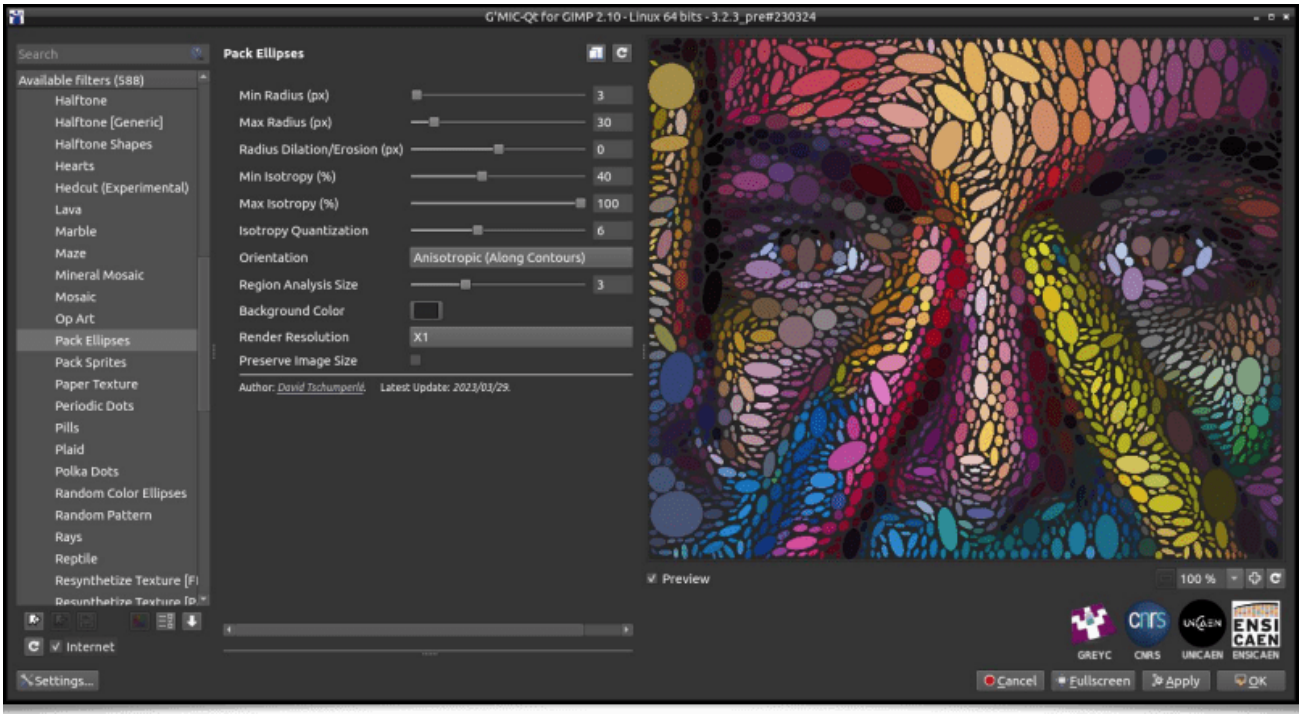

*Fig. 2.8. Le filtre **Patterns / Pack Ellipses**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

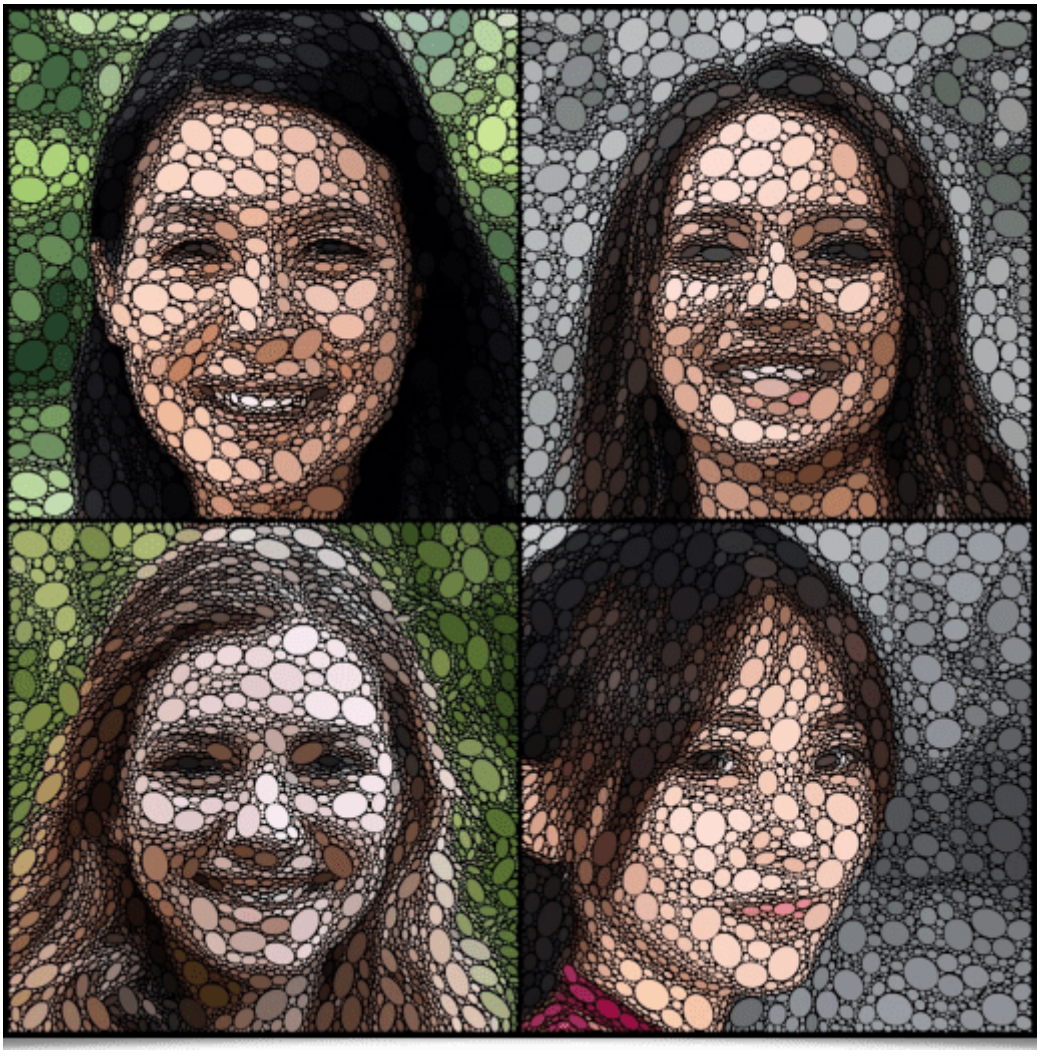

*Fig. 2.9. Application du filtre **Patterns / Pack Ellipses** sur différentes images de portrait.*

La vidéo ci-dessous illustre le comportement pas à pas de l'algorithme pour l'emboîtement de cercles colorés, afin de reconstituer l'image d'un portrait :

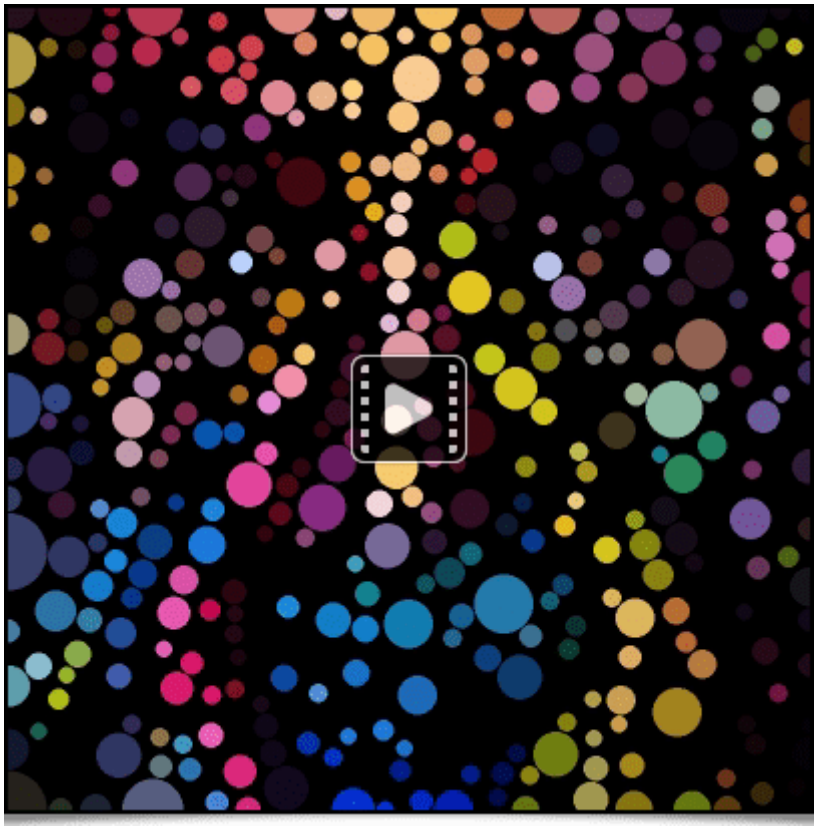

*Fig. 2.10. Les différentes étapes du filtre **Patterns / Pack Ellipses** décomposées en vidéo.*

- Toujours dans les effets de génération de textures et de motifs, signalons l'apparition d'un nouveau filtre de ***Halftoning***[W], nommé **Patterns / Halftone [Generic]**. Là encore, l'idée est de reconstituer une image d'entrée en empilant des motifs colorés de géométrie quelconque, par exemple de petits cercles :

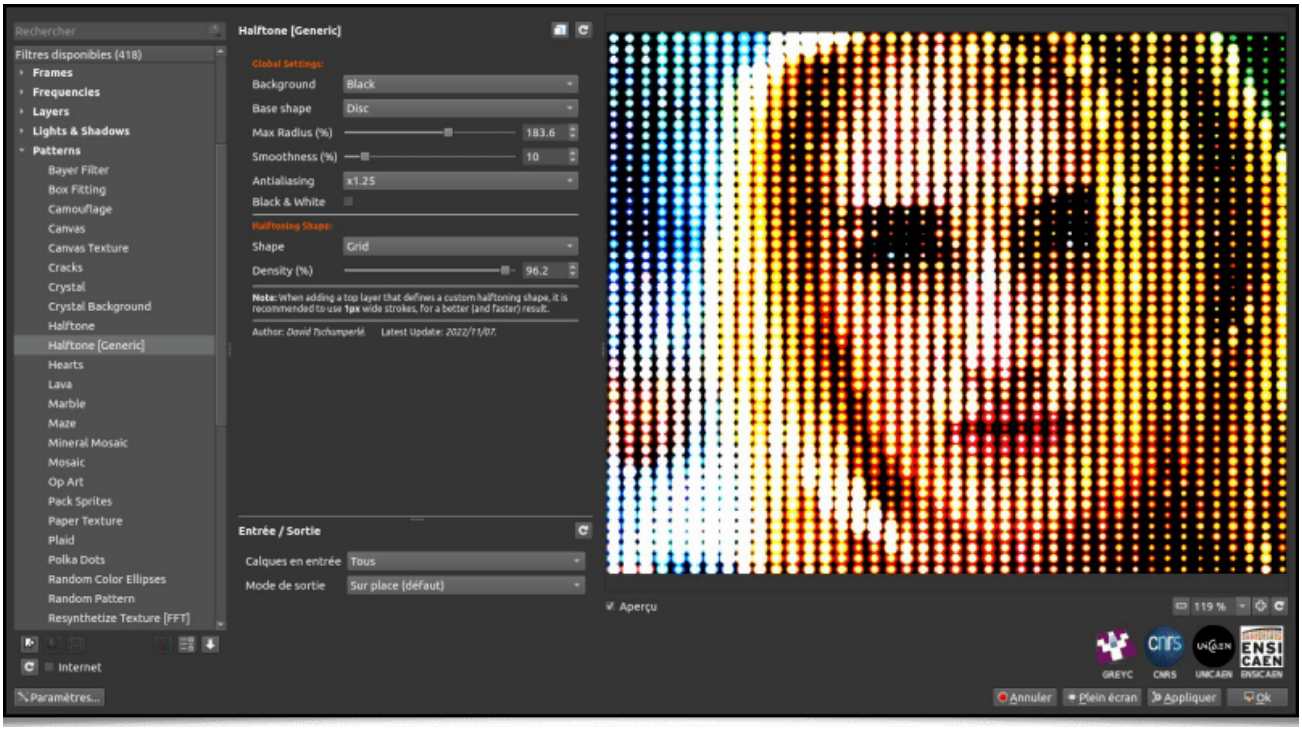

*Fig. 2.11. Le filtre **Patterns / Halftone [Generic]**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

Ou encore, une spirale :

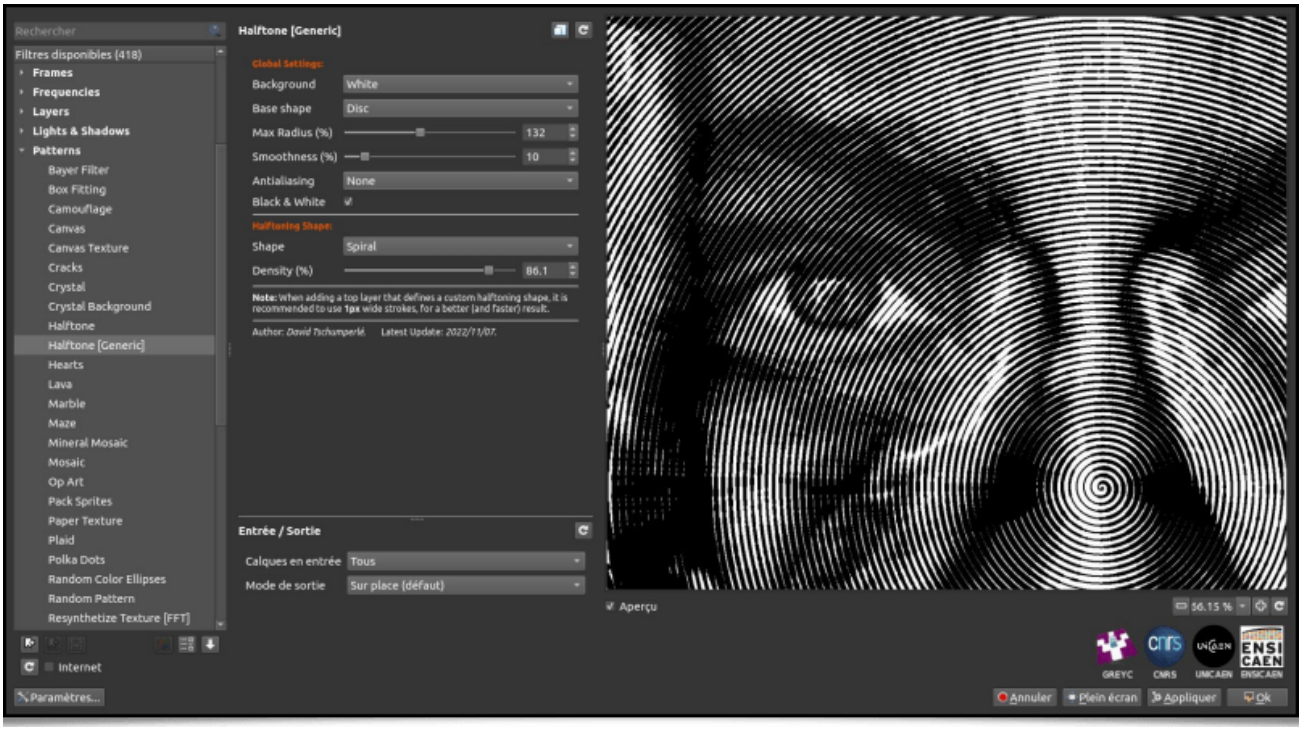

*Fig. 2.12. Filtre **Patterns** / **Halftone [Generic]** avec un motif en spirale.*

Le filtre propose même un mode spécial pour que l'utilisateur puisse fournir son motif de *Halftoning* personnalisé, dans un calque séparé :

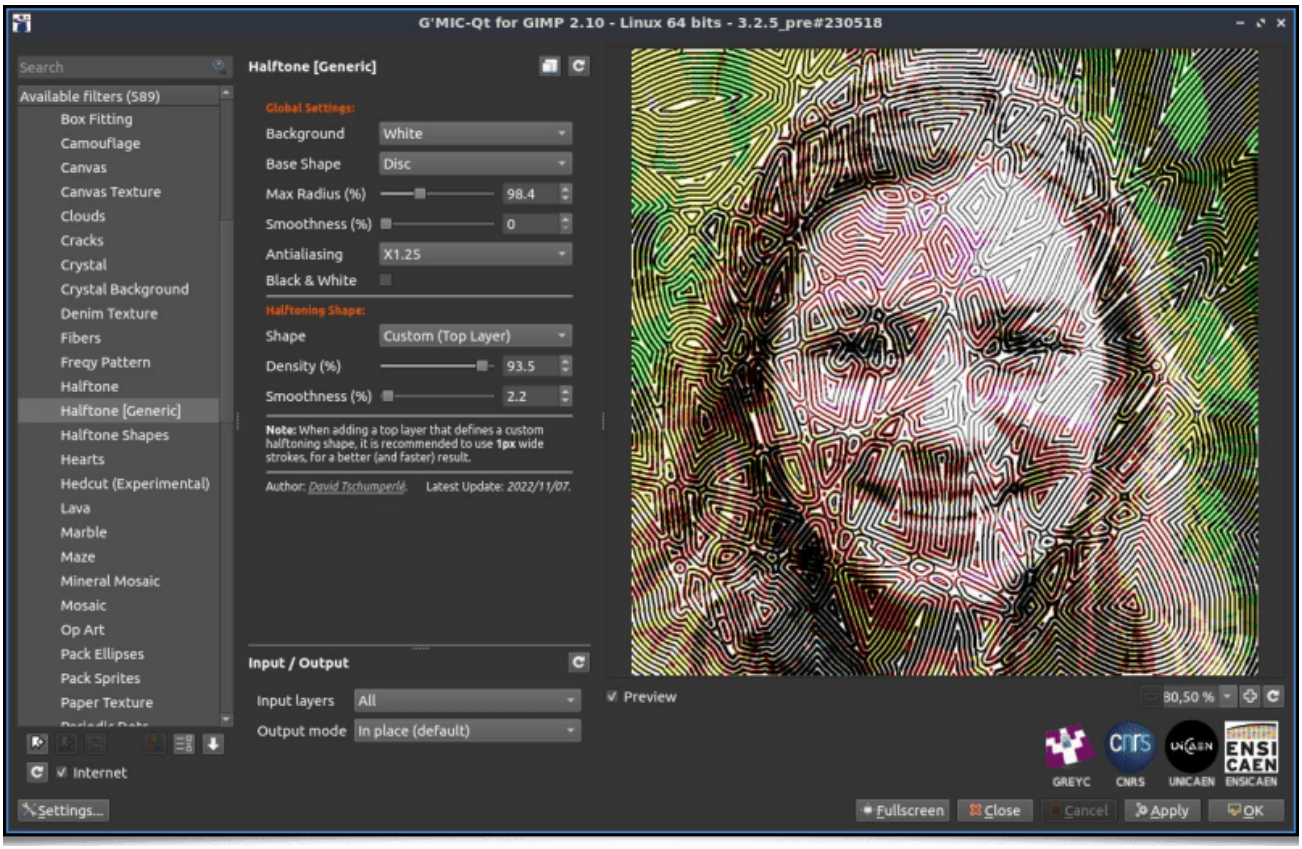

*Fig. 2.13. Filtre **Patterns** / **Halftone [Generic]** avec un motif personnalisé.*

D'un point de vue algorithmique, l'idée est d'éroder ou dilater localement le motif donné en paramètre du filtre, pour encoder au mieux le niveau de gris de chacun des pixels de l'image d'entrée.

- Le filtre suivant a une histoire amusante : étant abonné au compte *Twitter* de l'artiste **Memo Akten**, je suis tombé un jour sur **ce tweet** qui décrit un algorithme d'art génératif que Memo a imaginé (mais pas implémenté). Ce fut une bonne occasion d'essayer de l'implémenter en langage *G'MIC*, juste pour le plaisir d'expérimenter ! Une fois cela réalisé, créer un filtre utilisable dans le greffon *G'MIC-Qt* allait de soi. Il en

résulte le filtre **Rendering / Algorithm A**, qui crée des illustrations abstraites dans un esprit très « Mondrianesque[W] ».

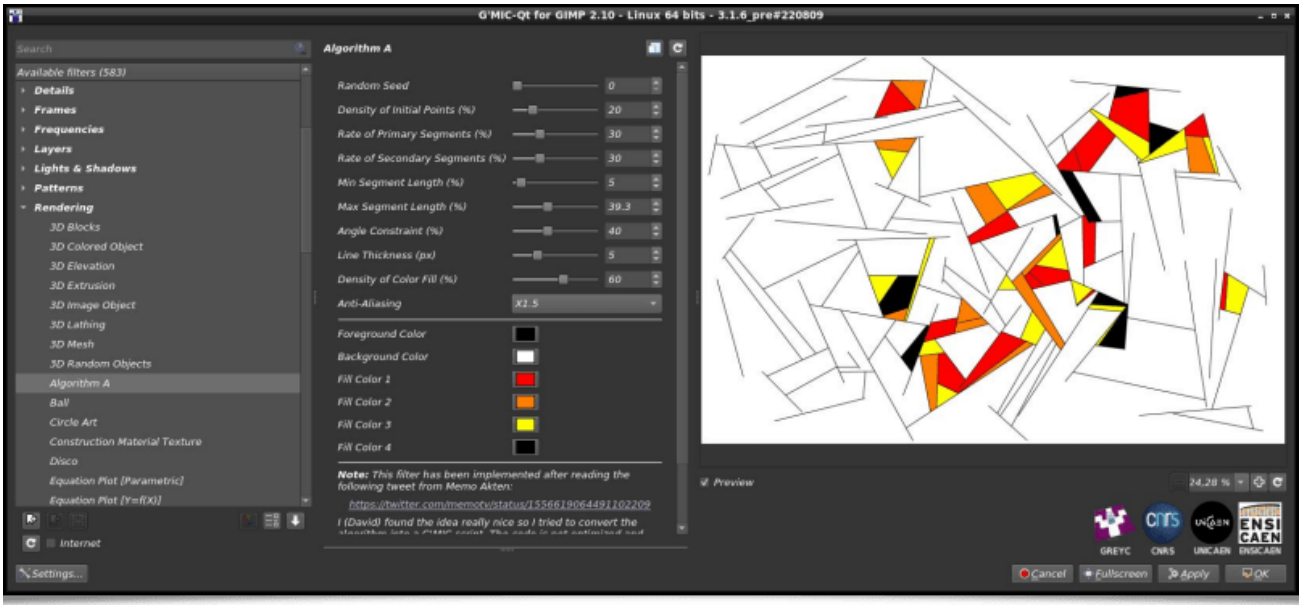

*Fig. 2.14. Le filtre **Patterns / Algorithm A**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

La génération des images repose largement sur le tirage de nombres aléatoires. D'une simple ligne de commande, on peut donc facilement produire plusieurs œuvres différentes à la suite :

```
$ gmic repeat 6 { 500,500,1,3 fx_memoakten_algorithm_a[-1] '$>',20,30,30,2,50,10,50,40,3,60,1,0,0,0,25
```

ce qui synthétise l'image suivante :

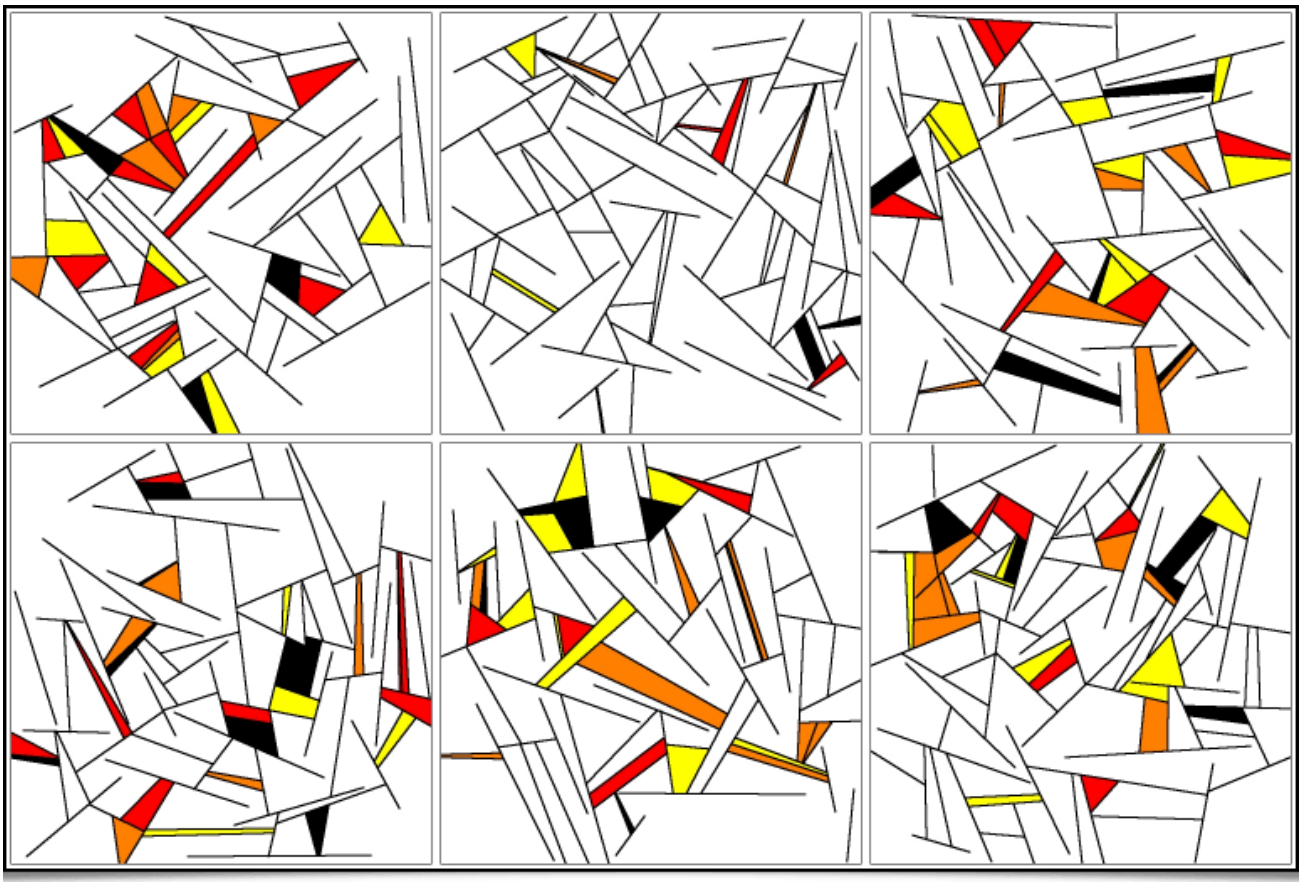

*Fig. 2.15. Patchwork d'« œuvres d'art », produites par le filtre **Patterns / Algorithm A**.*

- Toujours afin de produire des images bizarres et abstraites, évoquons l'apparition du filtre **Arrays & Tiles / Shuffle Patches**, qui va décomposer une image d'entrée sous la forme d'un tableau d'imagettes (« *patchs* »), et mélanger spatialement ces *patchs* avant de les recoller pour produire l'image résultat. Dif-

férentes options sont proposées, permettant la rotation aléatoire des *patchs*, ou le recollage de *patchs* qui se superposent.

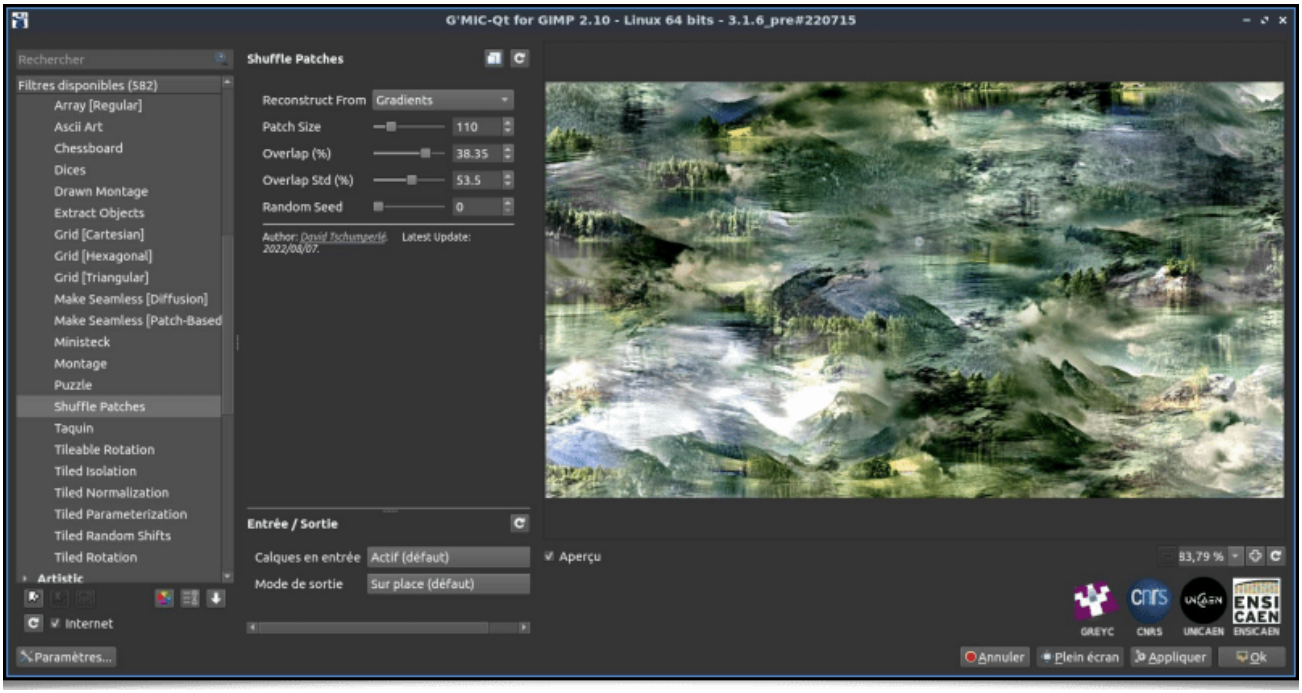

*Fig. 2.16. Le filtre **Arrays & Tiles / Shuffle Patches**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

On obtient ainsi une image qui ressemble à un collage de différentes parties de l'image d'origine, avec des couleurs globalement similaires, mais où l'on perd l'ordre naturel des structures.

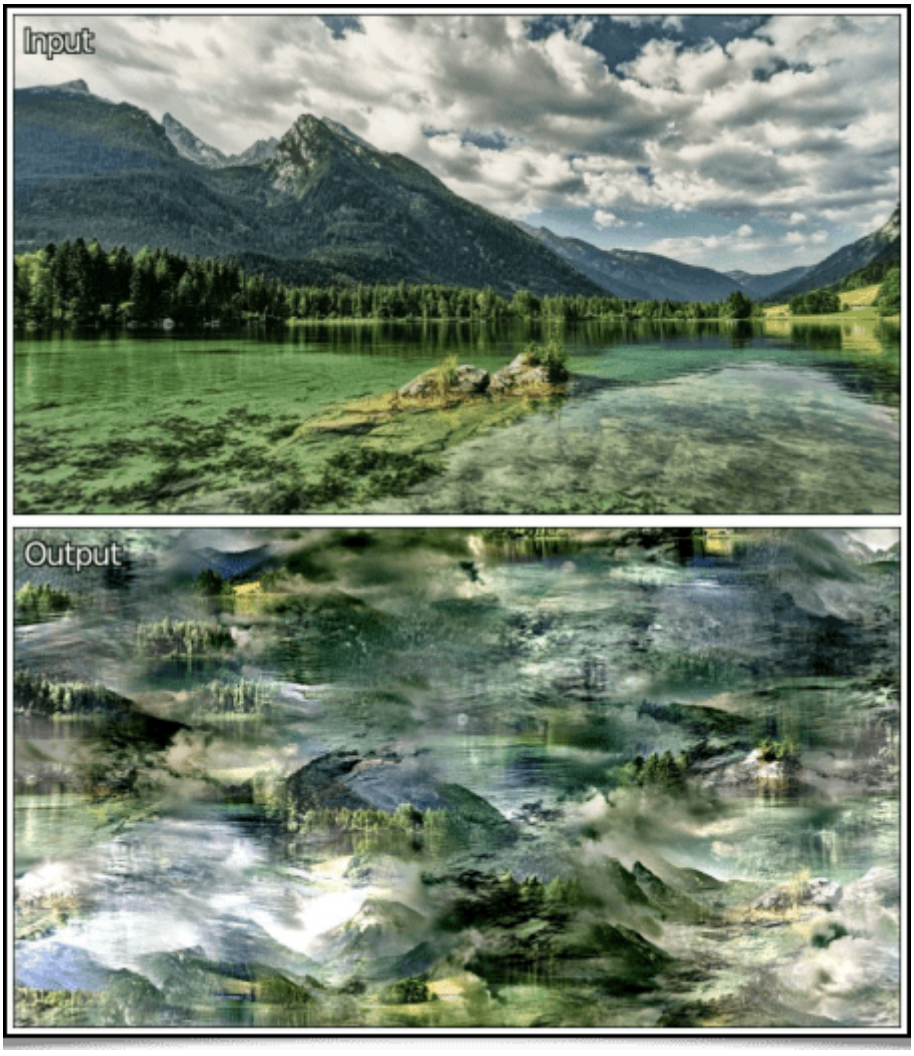

*Fig. 2.17. Effet du filtre **Arrays & Tiles / Shuffle Patches** sur une image de paysage.*

Ici encore, nous pouvons appliquer ce filtre sur toutes les *frames* d'une vidéo, illustré avec l'exemple ci-dessous (vous aurez bien sûr reconnu le court-métrage ***Big Buck Bunny***[W] de la **fondation *Blender***[W]).

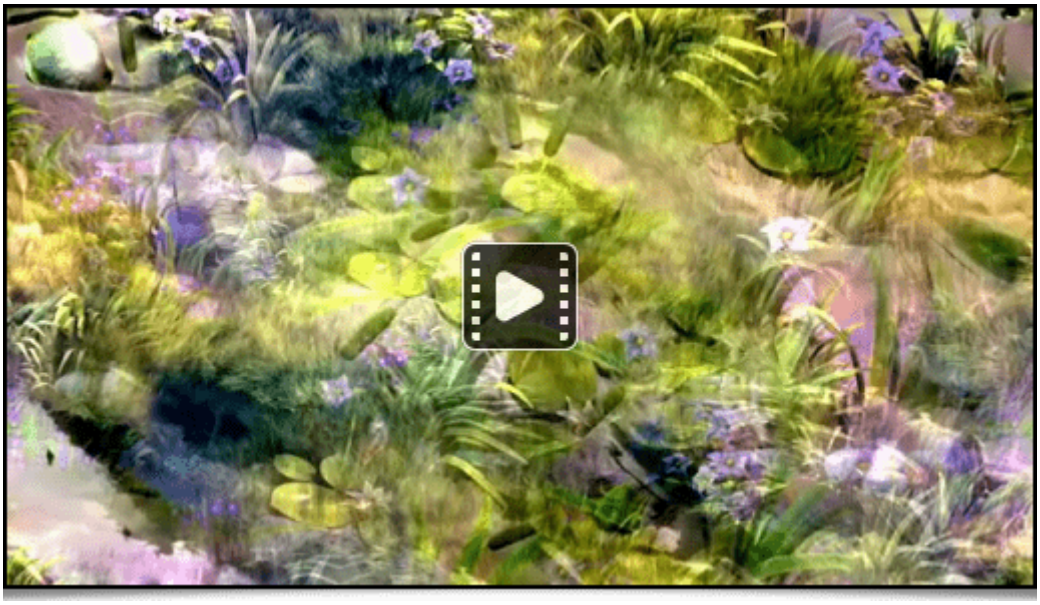

*Fig. 2.18. Le filtre **Arrays & Tiles / Shuffle Patches** appliqué sur la vidéo **Big Buck Bunny**[W] de la fondation Blender.*

- Et pour clôre cette section sur les effets d'abstraction d'images, de *Glitch Art* et de génération de motifs, voici le filtre **Patterns / Pills**, qui crée une texture périodique ressemblant à un empilement de « pilules » tournées de 90° les unes par rapport aux autres.

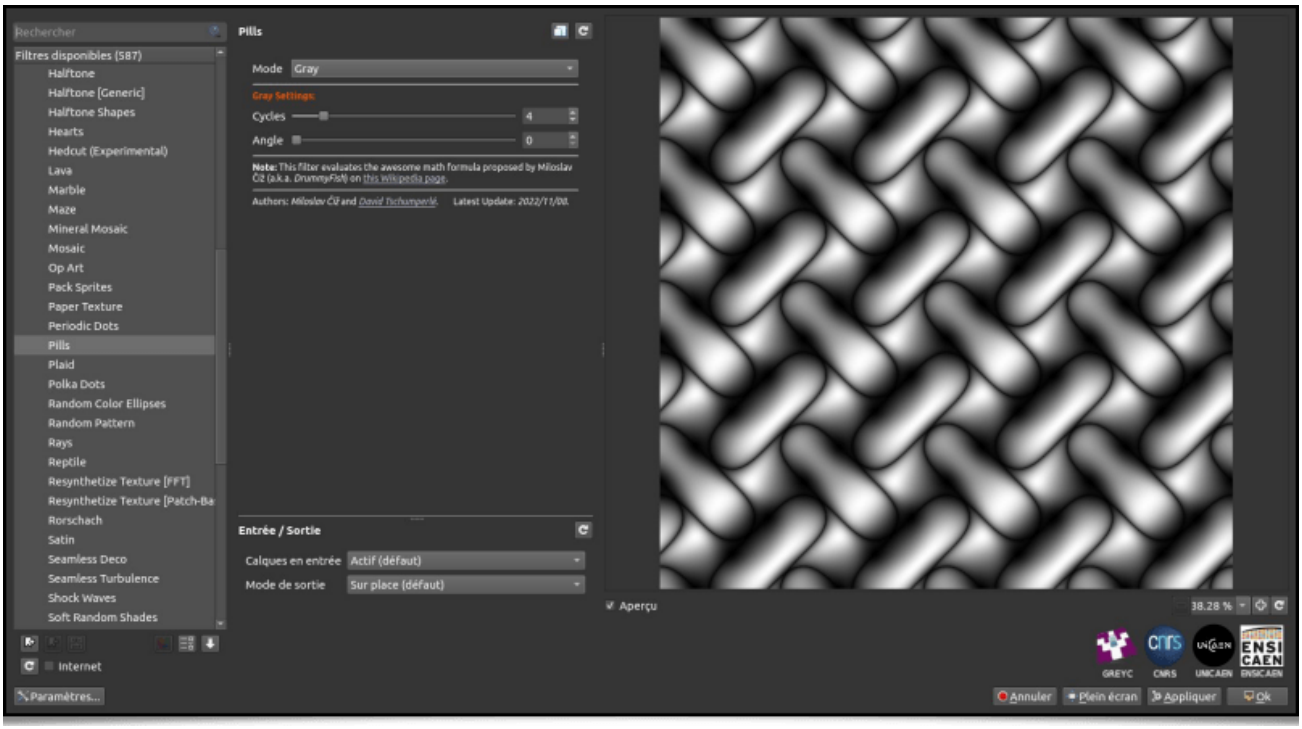

*Fig. 2.19. Le filtre **Patterns / Pills**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

Rien de très compliqué : ce filtre est une implémentation directe de la formule mathématique

$$\sqrt{|\sin(x+\cos(y+\sin(x+\cos(y))))\sin(y+\cos(x+\sin(y+\cos(x))))|}$$

Cette jolie formule a été imaginée par **Miloslav Čížž**[W], et décrite sur **cette page**. Il était tentant d'en faire un nouveau filtre accessible à tout le monde !

Notons que nous pouvons produire cette même image de base, directement à partir de la formule initiale, en lançant encore une fois `gmic` en ligne de commande :

```
$ gmic 600,600,1,1,"X = x*30/w; Y = y*30/h; sqrt(abs(sin(X + cos(Y + sin(X + cos(Y)))) * sin(Y + cos(X
```

Le filtre **Patterns / Pills** dans le greffon *G'MIC-Qt* autorise néanmoins quelques variations additionnelles, comme la possibilité de spécifier un angle de rotation ou de créer ces motifs indépendamment pour chacun des canaux *RGB* de l'image de sortie.

3. Nouveautés concernant le traitement des couleurs

3.1. Fonctionnalités pour les *LUTs* 3D

G'MIC est un logiciel de traitement d'images qui intègre nativement beaucoup de *LUTs 3D* couleur différentes (1045 à ce jour), en particulier grâce à un algorithme performant de compression de *LUTs* issu de nos travaux de recherche (décrit dans **une dépêche précédente**). Ces *LUTs* 3D couleur définissent des fonctions de transformation des couleurs d'une image, souvent pour lui donner une ambiance particulière. Récemment, de nouvelles commandes pour faciliter la visualisation et la création de *LUTs* 3D couleur ont été ajoutées à *G'MIC* :

- La commande `display_clut` fait le rendu 3D d'une *LUT* couleur, ce qui permet de visualiser la transformation *RGB* → *RGB* qu'elle représente. Par exemple, la commande :

```
$ gmic clut summer clut spy29 display_clut 400 text_outline[0] Summer text_outline[1] "Spy 29"
```

affichera :

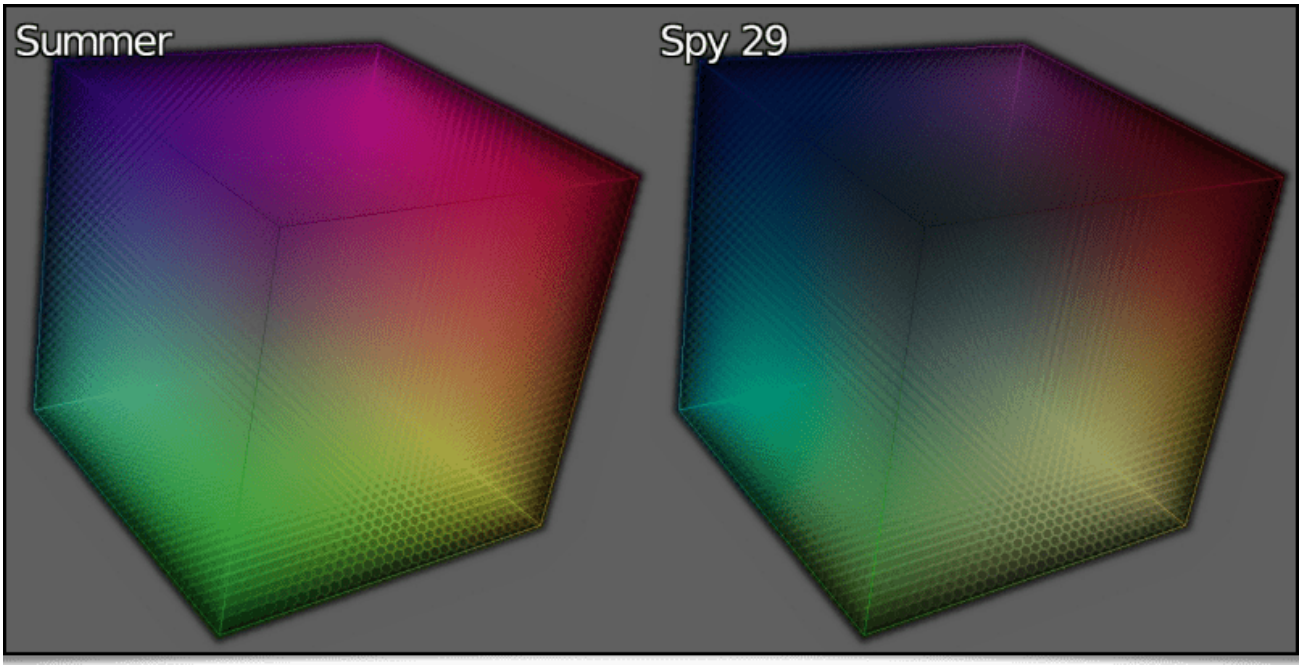

Fig. 3.1.1. La commande `display_clut` *affiche une vue 3D d'une _LUT* couleur._

- La commande `random_clut`, quant à elle, produit une *LUT* 3D couleur aléatoire, possédant certaines propriétés de continuité des couleurs. Par exemple, la commande suivante :

```
$ gmic sample colorful resize2dx 320 repeat 4 { random_clut +map_clut[0] [-1] display_clut[-2] 320 to_
```

synthétisera une image du style de celle ci-dessous :

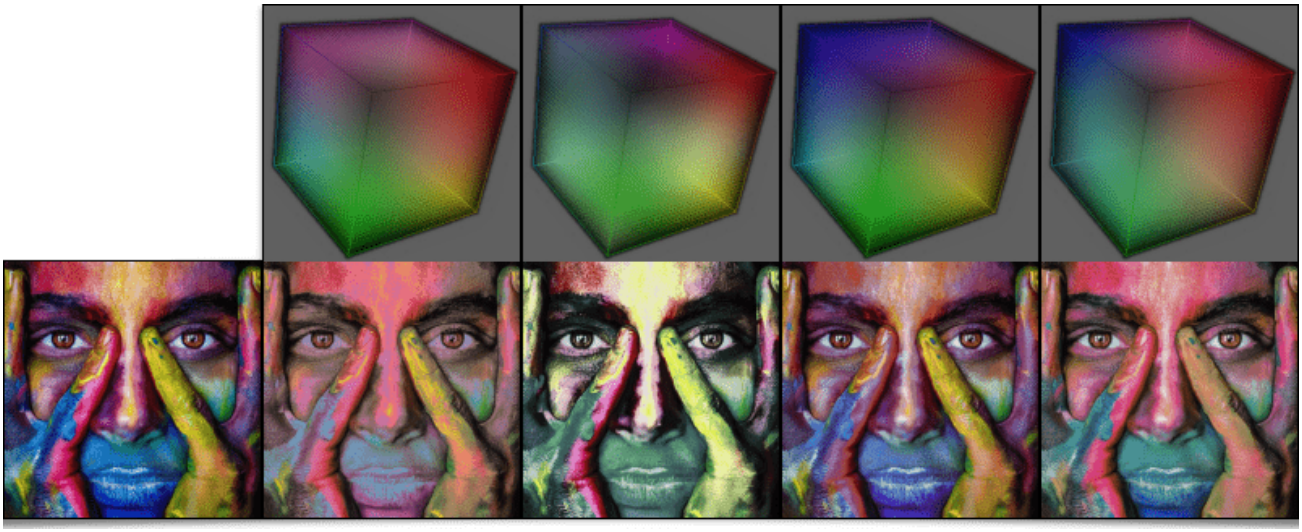

Fig. 3.1.2. Différentes _LUTs 3D couleur aléatoires, obtenues via la commande `random_clut`, et appliquées sur une image couleur._

3.2. Nouveaux filtres couleurs pour le greffon *G'MIC-QT*.

- Assez logiquement, la commande `random_clut` est à la base de l'implémentation du nouveau filtre **Colors / Random Color Transformation**, apparu dans le greffon *G'MIC-Qt*, qui applique une transformation colorimétrique aléatoire sur une image d'entrée.

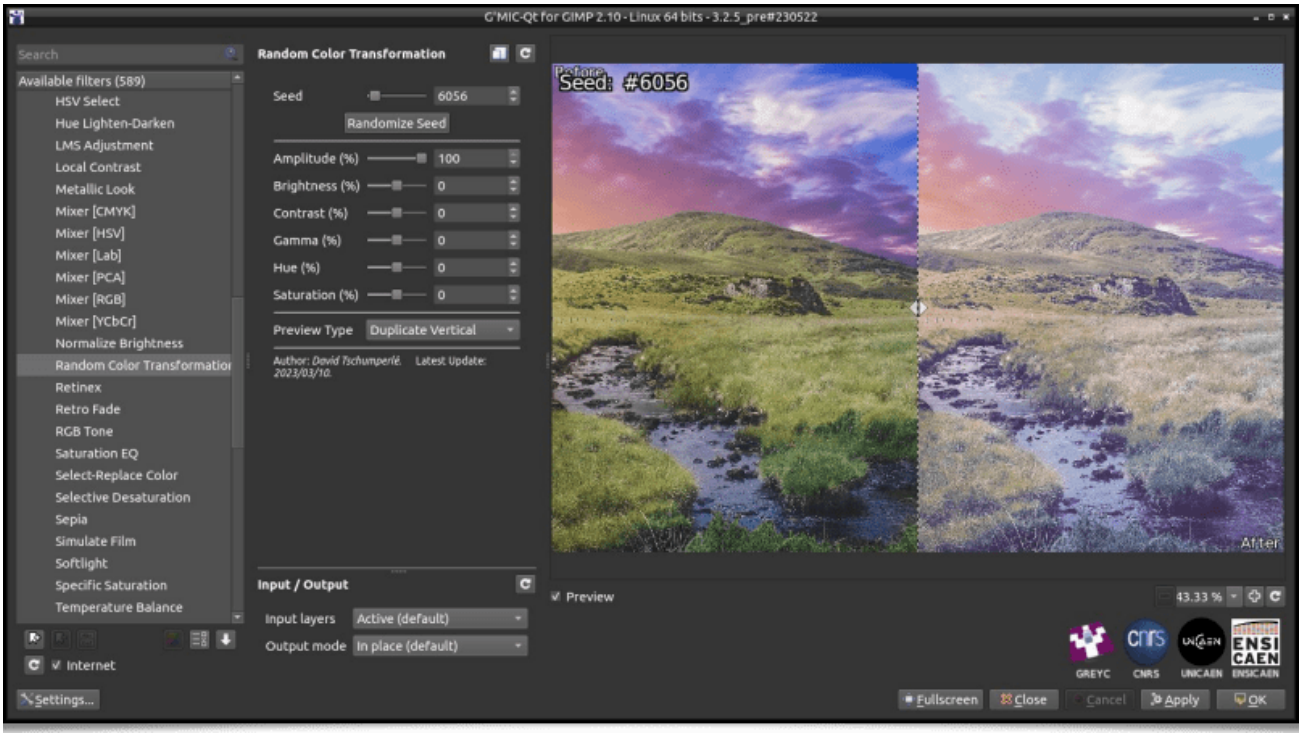

*Fig. 3.2.1. Le filtre **Colors / Random Color Transformation**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

- Pour rester dans le domaine des *LUTs* 3D couleur, mentionnons l'apparition du filtre **Colors / Apply From CLUT Set**, qui permet de transformer une image couleur en lui appliquant l'une des *LUTs* 3D définies dans un *pack* de *LUTs*, stocké lui-même sous la forme d'un fichier d'extension `.gmz`.

Quelques explications sont nécessaires : le format de fichier `.gmz` est implémenté et utilisé par *G'MIC* pour la sérialisation et la sauvegarde de données binaires génériques compressées.
Ainsi, comment produire un fichier `.gmz` stockant un ensemble de *LUTs* 3D couleur compressées, pour alimenter le filtre **Colors / Random Color Transformation** ? Prenons l'exemple concret du *pack* de 10 *LUTs* proposé gracieusement **sur cette page web**. Ces *LUTs* sont distribuées en format `.cube`, le format le plus répandu pour le stockage des *LUTs* 3D couleur. Ces 10 fichiers occupent **8.7 Mo** sur le disque.

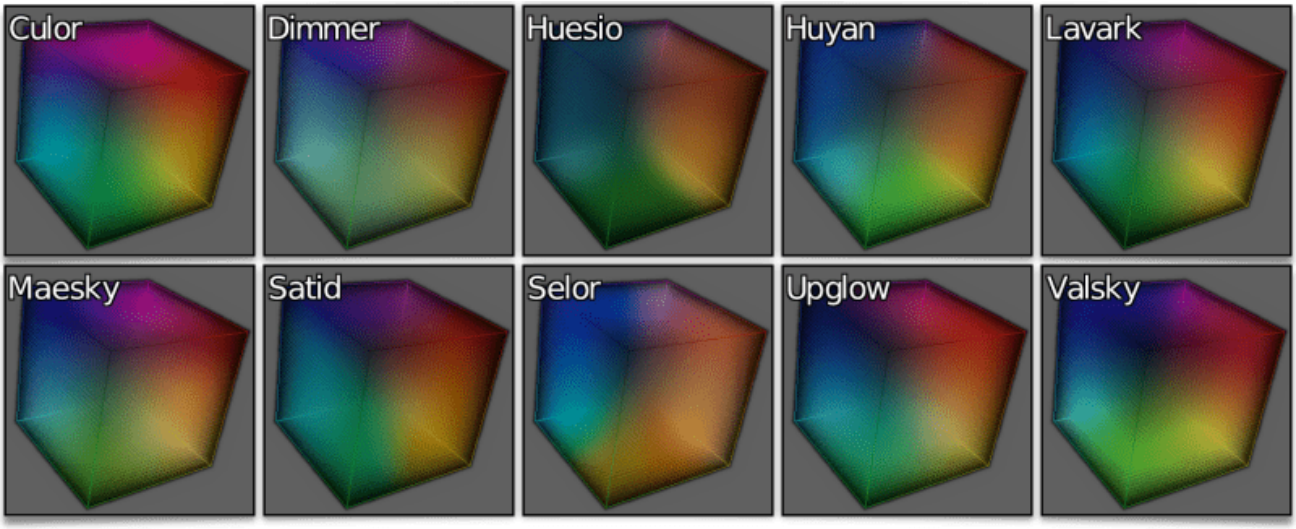

Fig. 3.2.2. Visualisation des 10 LUTs 3D couleur de notre cas d'exemple.

La commande suivante les compresse (avec des pertes visuellement imperceptibles) grâce à l'algorithme de compression de *LUTs* de *G'MIC*, en un fichier `clut_pack.gmz` de **156 Ko**. Attention, ce processus de compression est long (plusieurs dizaines de minutes) !

```
$ gmic *.cube compress_clut , output clut_pack.gmz
```

Une fois ce fichier de *pack* de *LUTs* généré, ces 10 transformations couleur sont disponibles, via le filtre **Colors / Apply From CLUT Set**, en lui spécifiant le fichier `clut_pack.gmz` en paramètre, comme illustré ci-dessous.

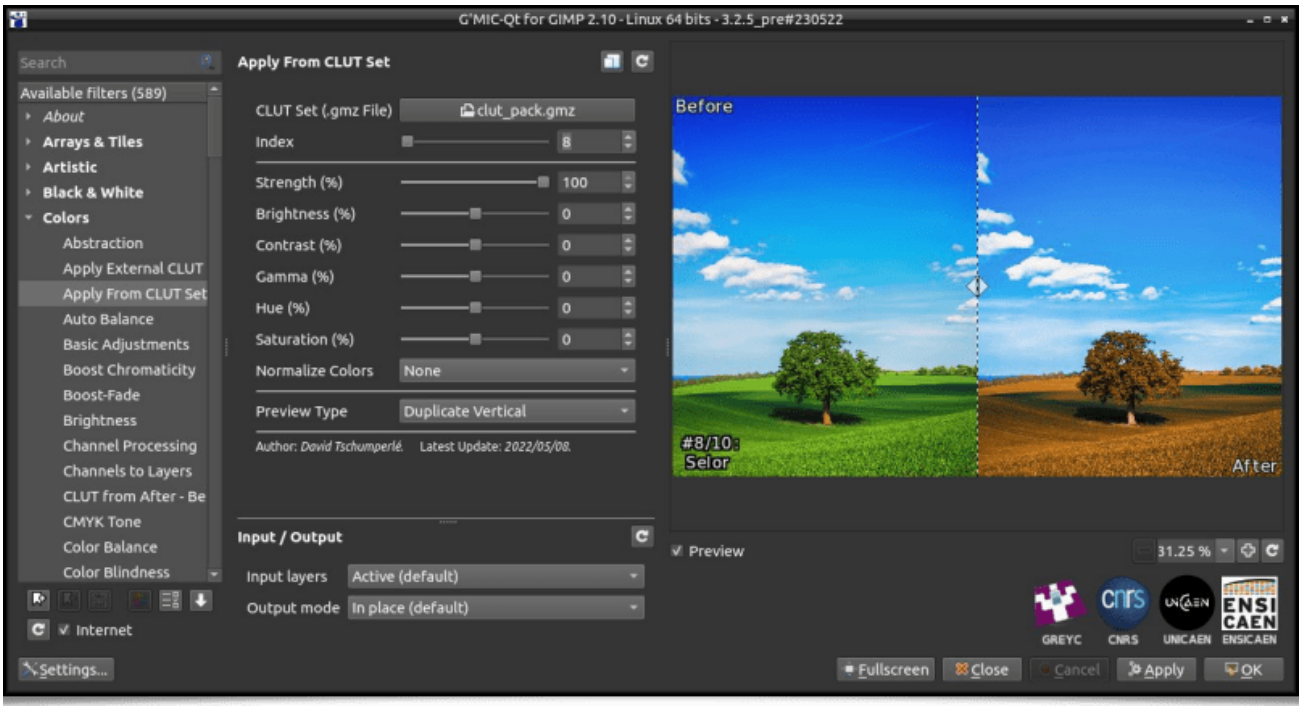

*Fig. 3.2.3. Le filtre **Colors / Apply From CLUT Set**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

Voilà donc un filtre évitant de stocker des *packs* de *LUTs* 3D couleur de plusieurs méga-octets sur son disque !

- Toujours dans le thème des transformations colorimétriques, voici le filtre récent **Colors / Vibrance**, qui rend les couleurs de vos images toujours plus chatoyantes. Ce n'est pas le premier filtre de ce type disponible dans *G'MIC*, mais on dispose ainsi d'une alternative aux algorithmes similaires déjà présents. Ce filtre émane de l'utilisateur ***Age***, qui participe occasionnellement aux discussions sur notre **forum**, hébergé par nos amis de **PIXLS.US** (dont **Pat David**, également contributeur au projet **GIMP**, est l'instigateur).

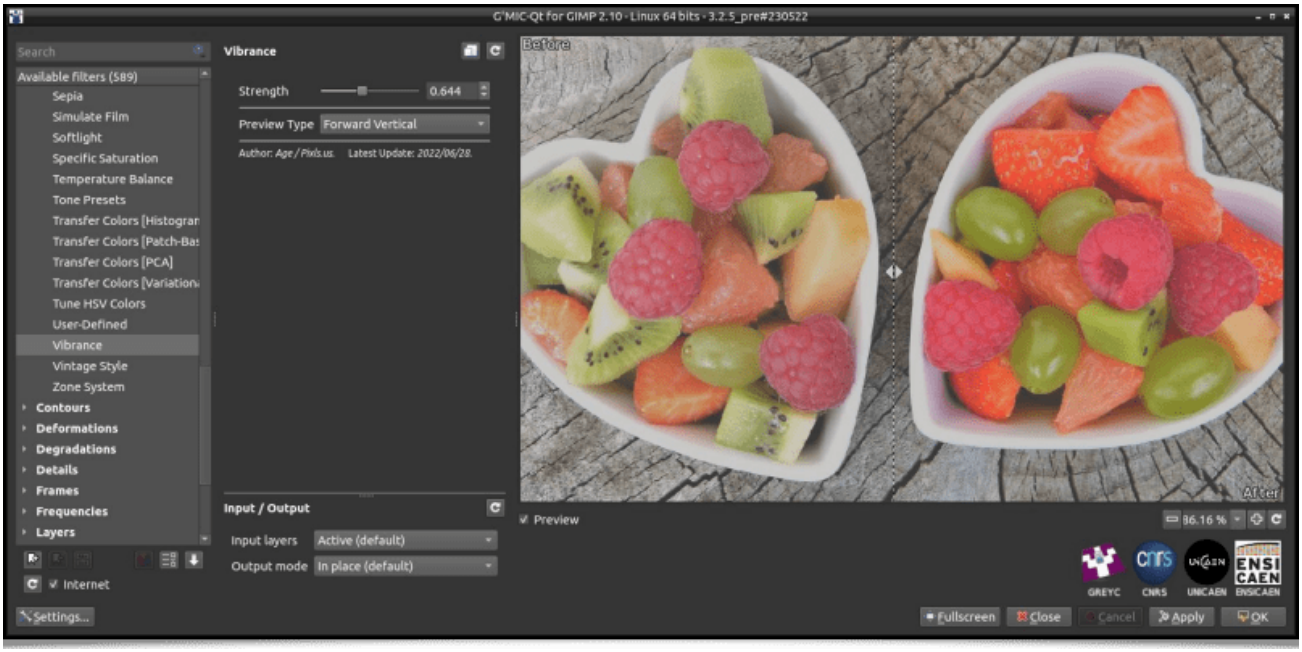

*Fig. 3.2.4. Le filtre **Colors / Vibrance**, tel qu'il apparaît dans le greffon G'MIC-Qt.*

3.3. Commandes `color2name` et `name2color`

Dernière nouveauté concernant les couleurs : les deux commandes `color2name` et `name2color`, qui convertissent un code couleur *RGB* en un nom de couleur (en anglais), et inversement. Un exemple d'utilisation est :

```
$ gmic 27,1,1,3 rand 0,255 split x foreach { color2name {^} resize 300,100 text_outline '${}',0.5~,0.5
```

Cette commande construit un tableau de couleurs aléatoires nommées, sous la forme d'une image telle que :

Fig. 3.3.1. Exemple d'utilisation de la commande `color2name` *pour nommer des couleurs aléatoires.*

Les associations entre les 881 noms de couleurs reconnues par ces commandes et leurs codes *RGB* respectifs ont été récupérées **de cette page Wikipédia** [W]. Ci-dessous, l'ensemble de ces 881 couleurs visualisées dans le cube *RGB* :

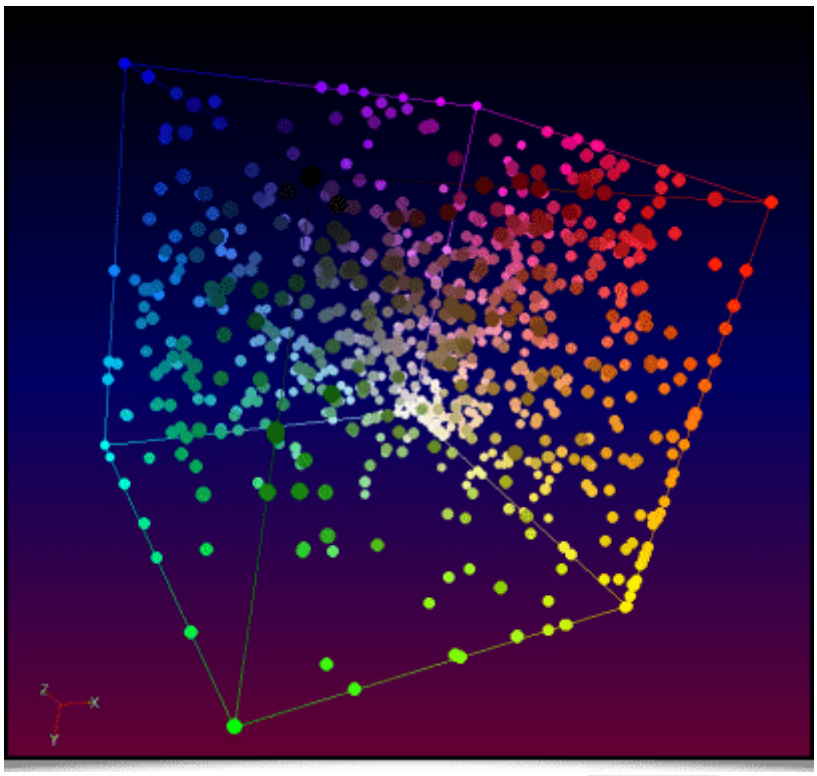

Fig. 3.3.2. Ensemble des couleurs nommées connues par la commande `color2name`.

4. Maillages 3D et ensembles de voxels

Saviez vous que *G'MIC* était aussi capable de manipuler des **objets 3D maillés** [W] (« *3D Mesh* » en anglais), en plus de gérer des images conventionnelles ? Et même si la modélisation et la visualisation 3D ne sont pas centrales dans les objectifs du projet, plusieurs ajouts intéressants ont été implémentés dans ce domaine.

4.1. Import d'objet en format *Wavefront*

Tout d'abord, *G'MIC* peut désormais importer des objets 3D en format `.obj` ***Wavefront*** [W], alors que seul l'export dans ce format était possible auparavant (export qui a d'ailleurs été amélioré). Toutes les caractéristiques du format `.obj` ne sont cependant pas prises en compte, mais l'import de la géométrie de l'objet, de ses couleurs et de ses textures fonctionnent en général. Ainsi, la commande :

```
$ gmic luigi3d.obj display3d
```

permet d'importer un objet 3D et de le visualiser dans une nouvelle fenêtre, comme l'illustre l'animation ci-dessous. Attention, le visualiseur intégré à *G'MIC* n'utilise pas les possibilités des *GPUs* pour l'accélération graphique. Le rendu peut donc être assez lent si le maillage comporte beaucoup de primitives (une piste d'amélioration pour le futur ?).

Fig. 4.1.1. Import et visualisation d'un maillage 3D texturé dans G'MIC.

Naturellement, nous avons intégré cette nouvelle fonctionnalité d'import de maillages 3D dans le greffon *G'MIC-Qt*, avec le nouveau filtre **Rendering / 3D Mesh**, qui permet d'importer un fichier `.obj` et d'en insérer un rendu 3D dans une image, comme l'illustre la vidéo ci-dessous :

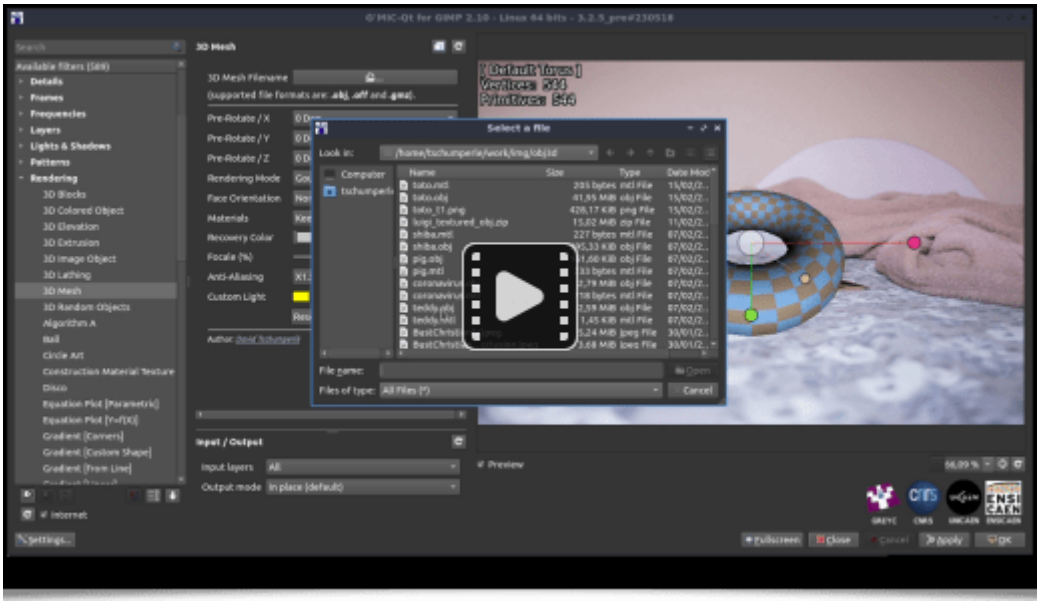

*Fig. 4.1.2 Le filtre **Rendering / 3D Mesh** en action, dans le greffon G'MIC-Qt.*

On l'utilisera typiquement pour importer un objet 3D du type que l'on souhaite dessiner, pour l'orienter dans l'espace, et s'en servir comme « guide » de dessin, soit en le redessinant complètement sur un calque supérieur, soit en utilisant un des nombreux filtres de *G'MIC*, par exemple pour en faire un rendu *cartoon* ou peinture.

4.2. Outils de transformation des maillages 3D.

Que faire d'autre, une fois un maillage 3D chargé en mémoire ? *G'MIC* s'est doté des fonctionnalités suivantes :

- L'extraction des textures de l'objet 3D, grâce à la nouvelle commande `extract_textures3d`. Les trois figures suivantes illustrent un cas d'utilisation, avec l'exemple d'un objet 3D maillé de chat dont la texture est extraite, transformée avec un filtre de stylisation (basé sur le modèle de l'estampe japonaise **La Grande Vague de Kanagawa** [W]), puis réappliquée sur le chat.

Fig. 4.2.1. Vue d'un objet 3D maillé representant un chat, avec sa texture.

Fig. 4.2.2. Extraction et stylisation des textures, via la commande `extract_textures3d`.

Fig. 4.2.3. Visualisation du maillage 3D original et de sa version stylisée.

- On peut également subdiviser les faces d'un objet 3D, grâce à la nouvelle commande `subdivide3d`.

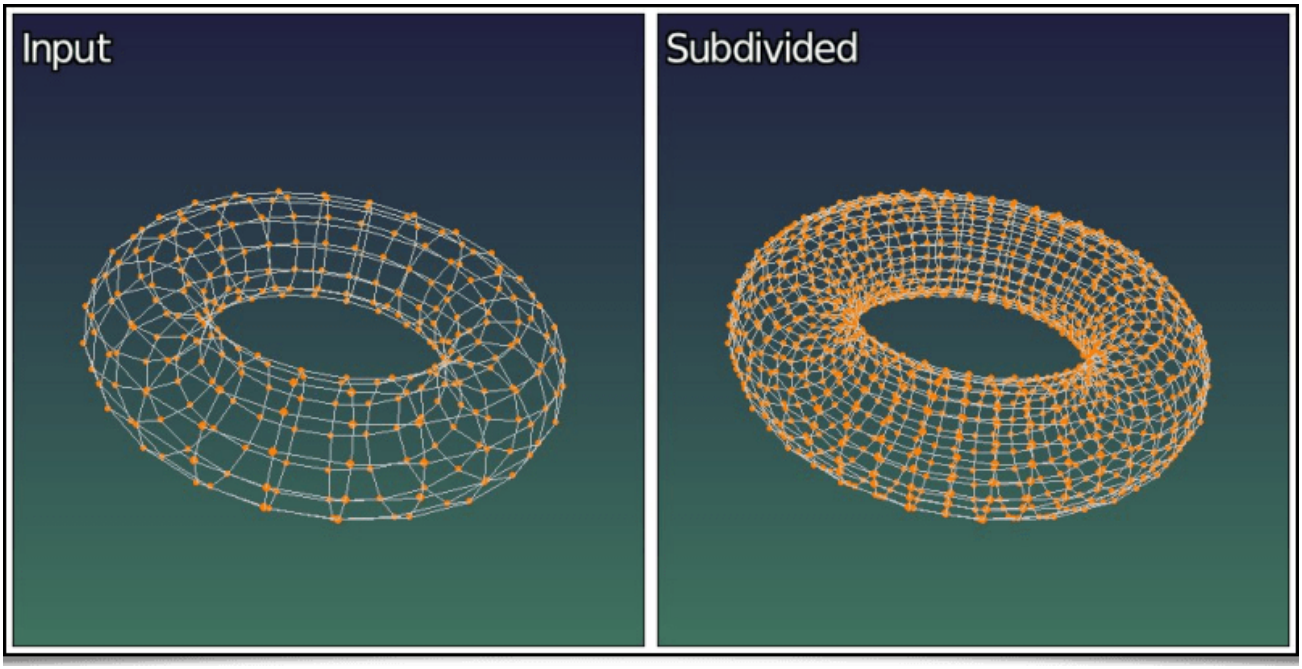

Fig. 4.2.4. Subdivision des faces d'un maillage 3D de tore, par la commande `subdivide3d`.

- On peut convertir un objet 3D **texturé** en objet uniquement **coloré**, avec la commande `primitives3d`. La commande suivante applique ce processus sur l'objet *Luigi3d* introduit précédemment, pour lui retirer sa texture et la remplacer par des faces colorées :

```
$ gmic luigi3d.obj primitives3d 2 output luigi3d_no_textures.obj
```

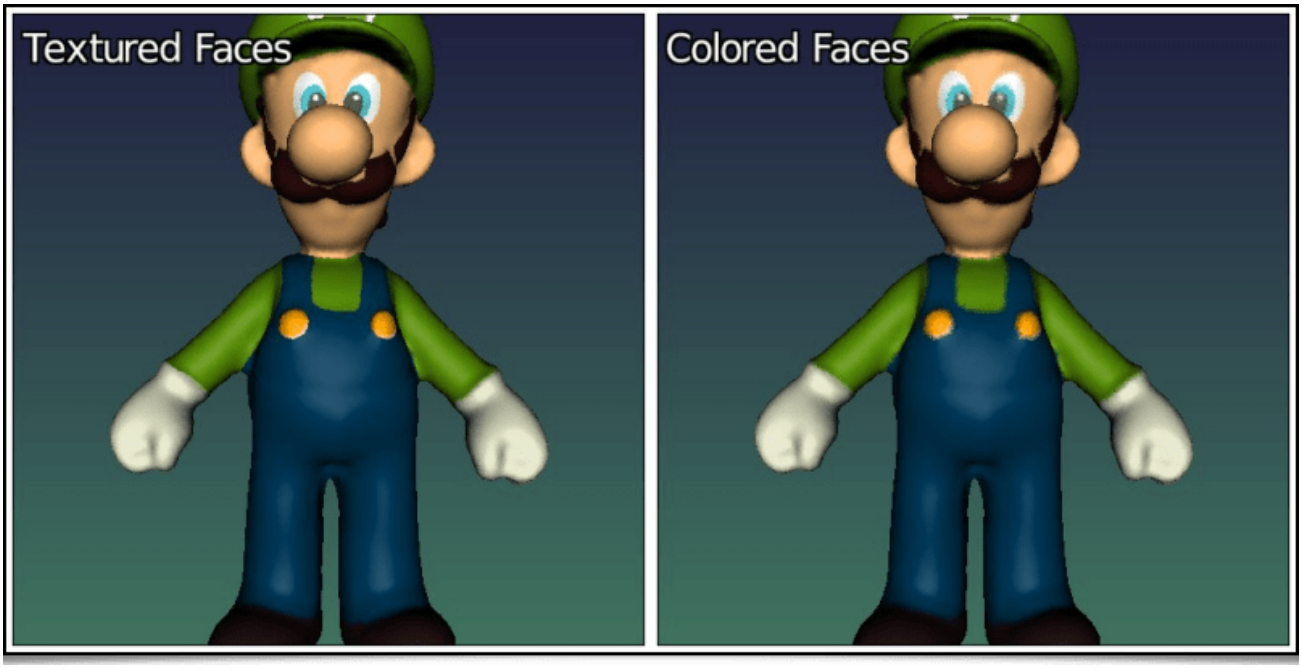

Fig. 4.2.5. Conversion des primitives texturées en primitives colorées uniquement, avec la commande `primitives3d`.

La couleur de chaque face est calculée comme la moyenne des couleurs à chacun des sommets composant la face. Pour les grandes faces, cela peut donc être très utile de subdiviser l'objet préalablement pour obtenir une résolution de texture colorée suffisante dans l'objet final (avec la commande `subdivide3d`).

- Enfin, on peut également convertir l'objet maillé 3D sous forme d'une image **volumique** contenant un ensemble de **voxels**, grâce à la nouvelle commande `voxelize3d`. Cette commande convertit toutes les primitives de base composant un objet 3D (points, faces, segments, sphères), sous la forme de primitives discrètes tracées dans l'image volumique. Par exemple, la commande :

```
$ gmic skull.obj voxelize3d 256,1,1 output skull_voxelized.tif display_voxels3d
```

va transformer le maillage du crane illustré ci-dessous, sous la forme d'une image volumique de voxels en couleurs, que l'on va pouvoir visualiser avec la nouvelle commande `display_voxels3d`. D'où un rendu très « **Minecraftien**[W] » (representé ci-dessous pour différentes résolutions de voxélisation) :

Fig. 4.2.6. Conversion d'un maillage 3D texturé sous forme d'images volumiques composé de voxels colorés, avec la commande `voxelize3d`.

Cette fonctionnalité servira, par exemple, aux personnes étudiant le domaine de la géométrie discrète, qui pourront facilement générer des objets discrets 3D complexes à partir de maillages (souvent plus faciles à produire que leurs équivalents discrets !). La vidéo ci-dessous illustre le rendu d'un objet 3D discret obtenu de cette façon :

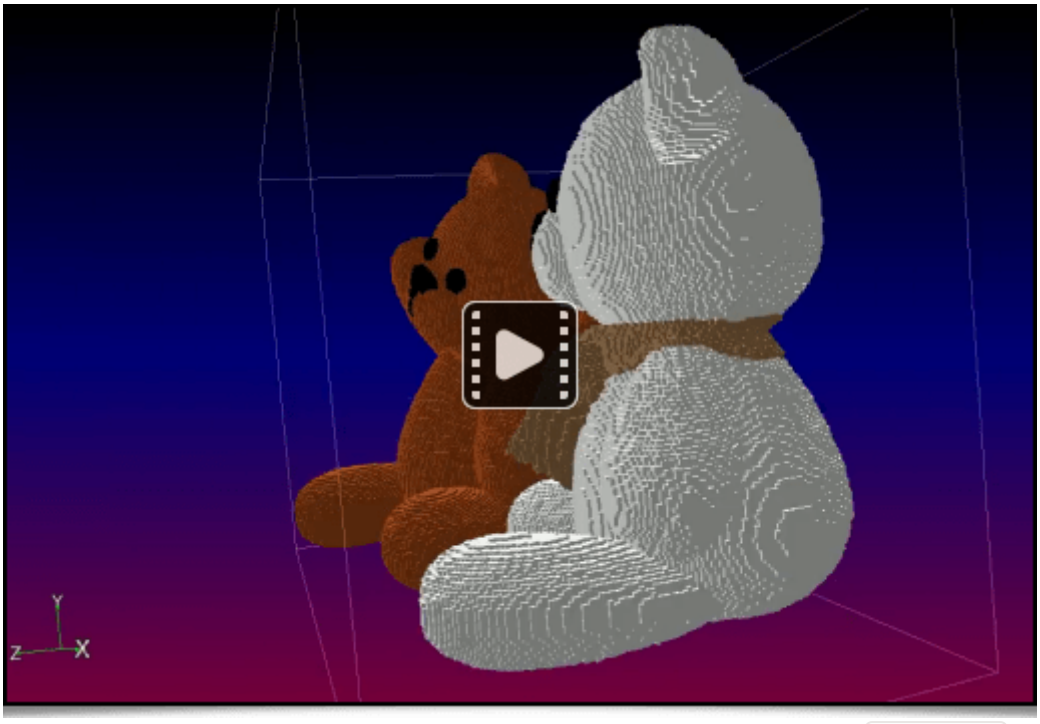

Fig. 4.2.7. Vidéo de visualisation d'un maillage 3D complexe voxélisé par la commande `voxelize3d`.

4.3. Outils de génération de maillages 3D

Pour conclure cette partie dédiée aux maillages 3D dans *G'MIC*, mentionnons l'apparition de quelques commandes récentes, en vrac, pour la génération procédurale d'objets 3D maillés :

- Les commandes `shape_menger` et `shape_mosely` produisent des représentation volumiques (images de voxels) des objets mathématiques fractals suivants : L'**éponge de Menger**[W] et le **flocon de Mosely**[W].

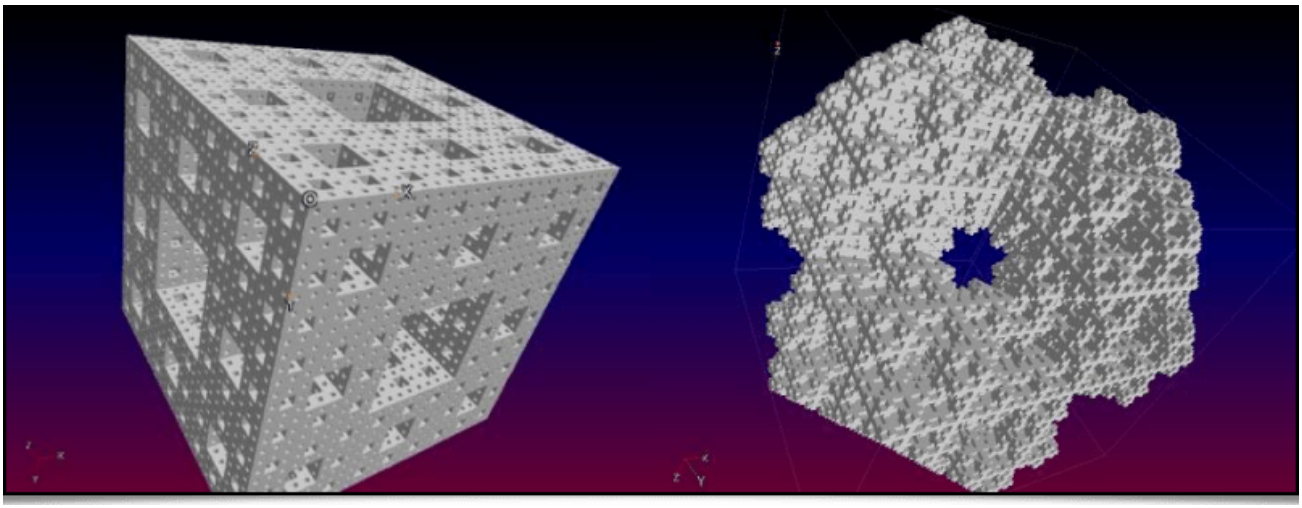

Fig. 4.3.1. Rendus 3D de l'éponge de Menger et du flocon de Mosely, crées avec les commandes `shape_menger` *et* `shape_mosely`.

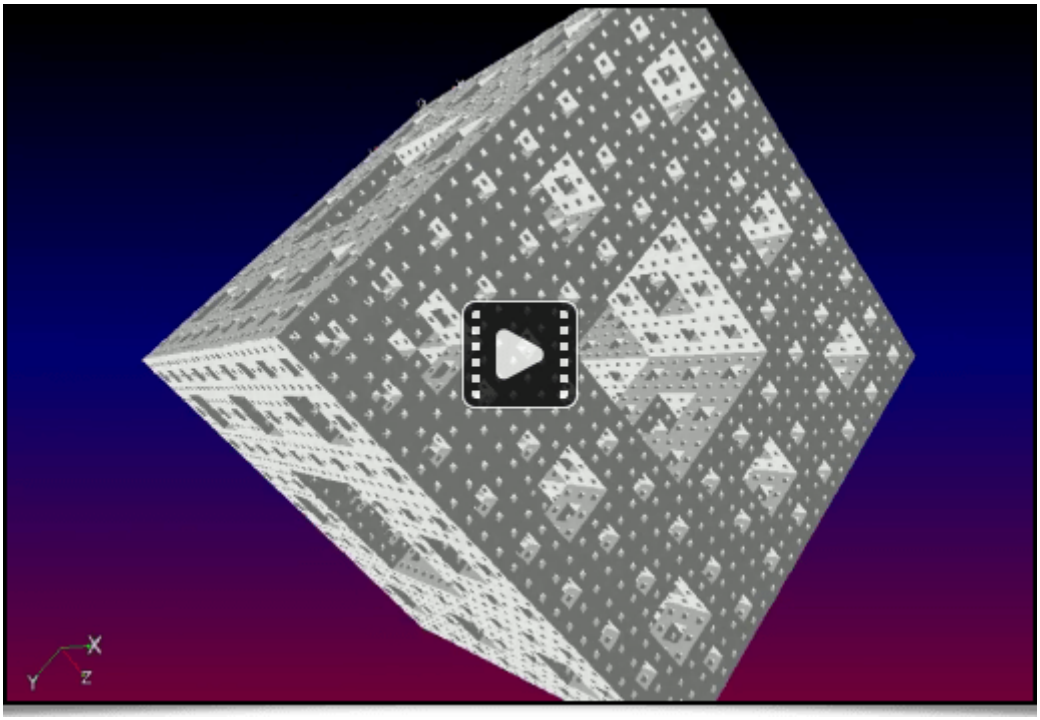

Fig. 4.3.2. Vidéo d'une éponge de Menger, rendue par G'MIC.

- La commande `chainring3d` produit un anneau 3D de tores colorés :

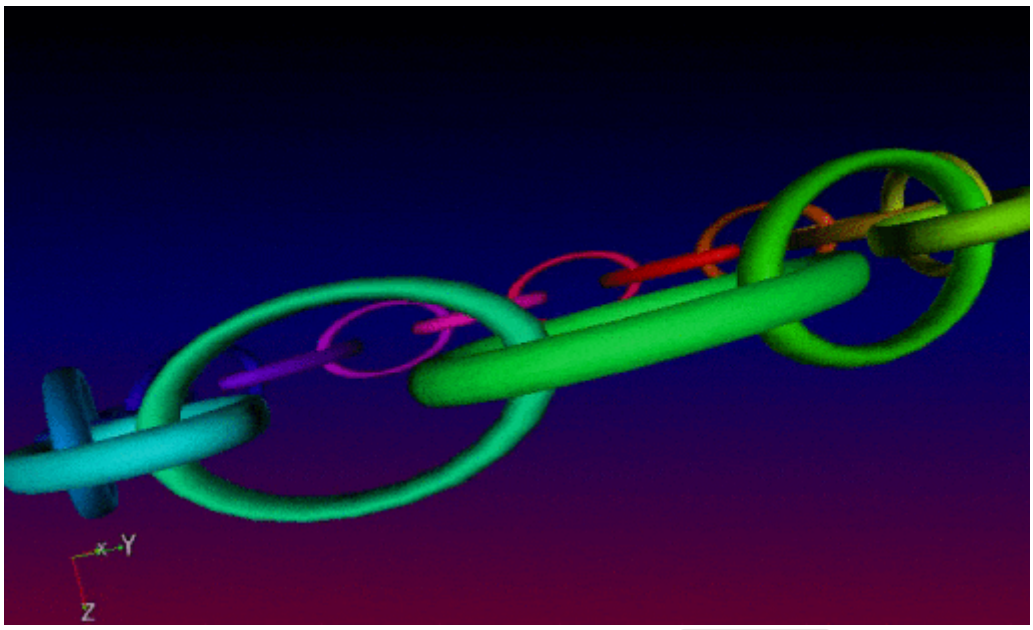

Fig. 4.3.3. Rendu d'un anneau 3D de tores colorés, avec la commande `chainring3d`.

- La commande `curve3d` génère le maillage 3D d'une courbe paramétrée `t → (x(t),y(t),z(t))` avec optionnellement une épaisseur de rayon `r(t)`, elle aussi paramétrée.

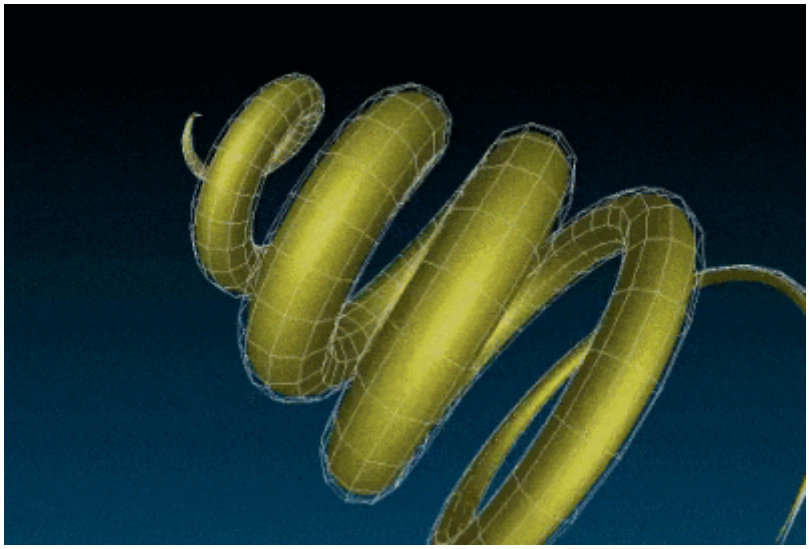

Fig. 4.3.4. Rendu d'une courbe 3D paramétrée, créé par la commande `curve3d`.

- La commande `sphere3d` peut maintenant produire des maillages 3D de sphères par trois méthodes différentes : 1. la subdivision d'isocahèdres, 2. la subdivision de cubes, et 3. la discrétisation angulaire en coordonnées sphériques. Les voici illustrées de gauche à droite, ci-dessous :

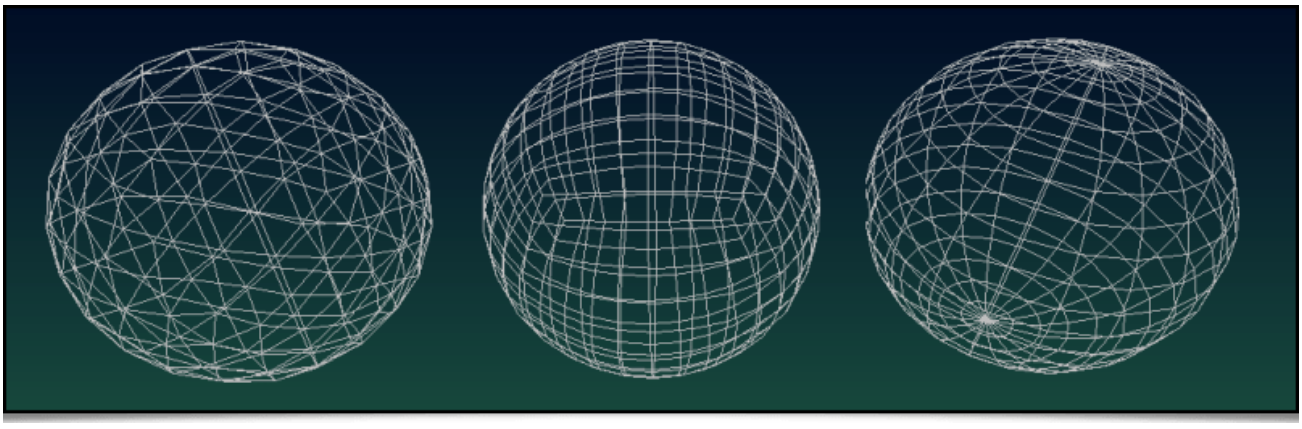

Fig. 4.3.5. Production de maillages 3D de sphères, avec trois algorithmes de maillage différents, via la commande `sphere3d`.

En pratique, toutes ces nouvelles commandes de création de maillages 3D peuvent s'insérer dans des *pipelines G'MIC* plus complexes, afin de construire des objets 3D sophistiqués de manière procédurale. Ces maillages pourront ensuite être exportés en fichier `.obj`. En voici l'illustration, avec la création d'un anneau de chaines récursif qui a été d'abord généré par *G'MIC* (utilisant d'ailleurs la commande `chainring3d` comme élément de base), puis importé dans ***Blender*** :

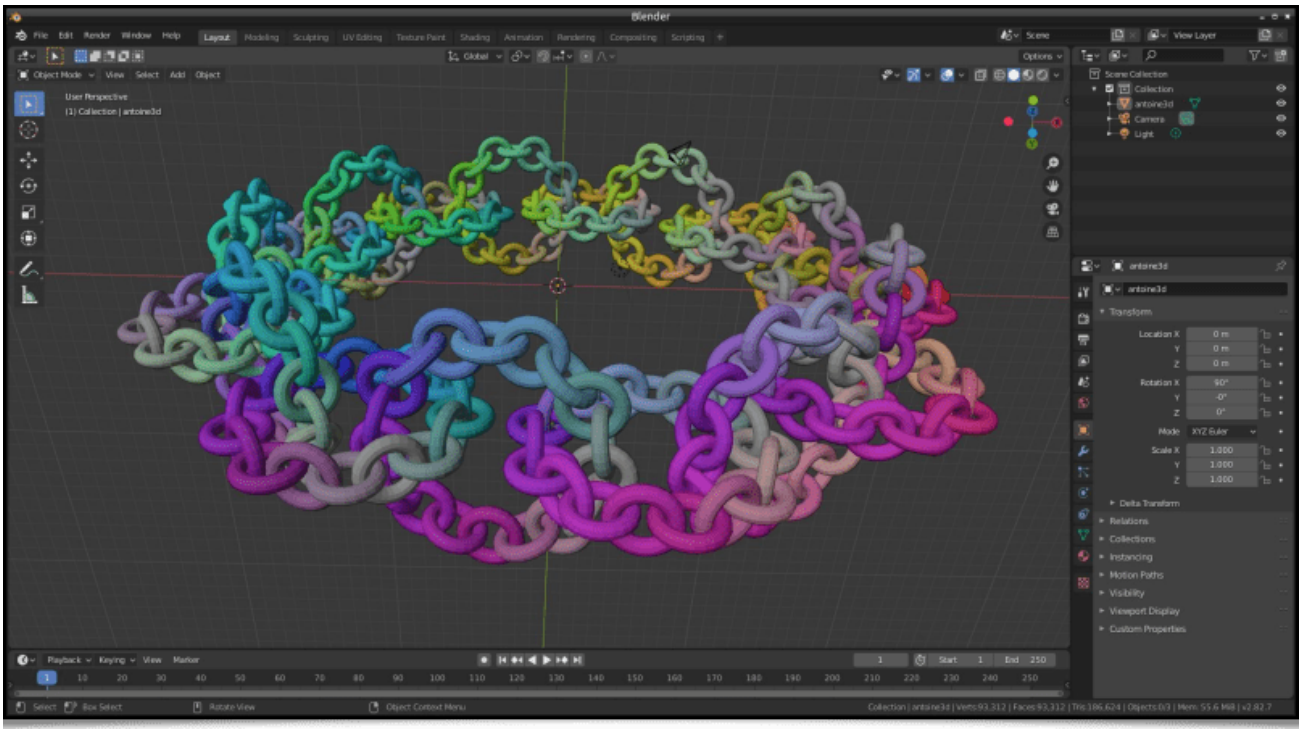

Fig. 4.3.6. Génération procédurale d'objet faite en G'MIC, puis importé dans Blender.

5. Autres nouveautés

Cette dernière section livre quelques autres informations liées au projet, sans ordre précis.

5.1. Amélioration diverses du greffon G'MIC-Qt

Beaucoup de travail a été accompli sur le code du greffon *G'MIC-Qt*, même si cela n'est pas forcément visible au premier abord. Citons en particulier :

- Des **optimisations importantes** du code qui améliorent le temps de démarrage du greffon : la fenêtre du greffon s'affiche plus rapidement, l'analyseur des filtres est plus efficace, notamment grâce à l'utilisation d'un cache binaire stockant les informations des filtres analysés après une mise à jour.
- Des **améliorations** concernant la **stabilité** du greffon. Il gère mieux les *threads* lancés par des filtres non terminés.
- Un changement du thème de l'interface, qui passe par défaut **en mode sombre** (« *Dark Mode* »).
- Le temps d'exécution d'un filtre apparaît maintenant lorsque l'on clique sur le bouton « *Apply* ».
- Un nouveau système de gestion des **sources externes de filtres** : il devient facile pour un développeur de partager ses filtres *G'MIC* personnalisés avec un utilisateur, en lui fournissant un fichier ou une URL pointant vers leur implémentation (à la manière des ***PPA*** pour les gestionnaires de paquets sous *Ubuntu*).

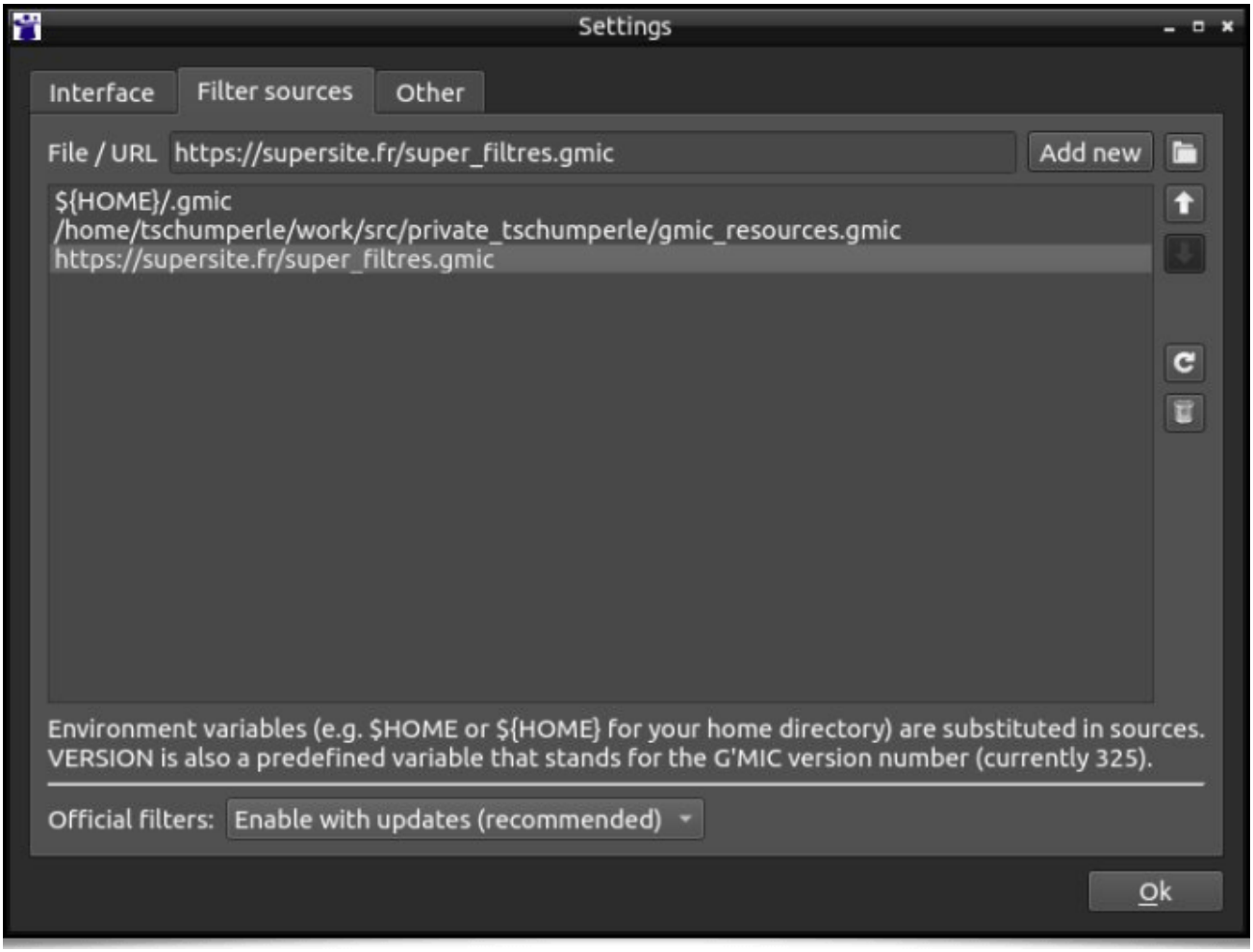

Fig. 5.1.1. Le nouveau système de gestion des sources de filtres externes dans le greffon _G'MIC-Qt._

- Le greffon offre un moyen aux filtres de conserver des données persistantes dans un cache en mémoire, lors d'appels consécutifs. Cela permet aux filtres ayant besoin de charger ou générer des données volumineuses, de ne le faire qu'une fois, et de les réutiliser lors des appels suivants. Ce système est par exemple utilisé par le filtre **Rendering / 3D Mesh** pour stocker l'objet 3D lu à partir d'un fichier.
- Le code du greffon a été modifié pour faciliter un future portage vers la version 6 de la bibliothèque *Qt*.
- Grâce au travail de **Nicholas Hayes**, le greffon *G'MIC-Qt* est maintenant disponible sur le ***Marketplace d'Adobe***. L'installation du greffon est donc maintenant facilitée pour les utilisateurs de **Photoshop**[W].
- Mentionnons enfin la mise à jour du greffon pour la dernière version de **Digikam** (la *8.0.0*), grâce au travail de **Gilles Caulier**. Une **documentation fournie** a été mise en ligne sur le site de *Digikam*.

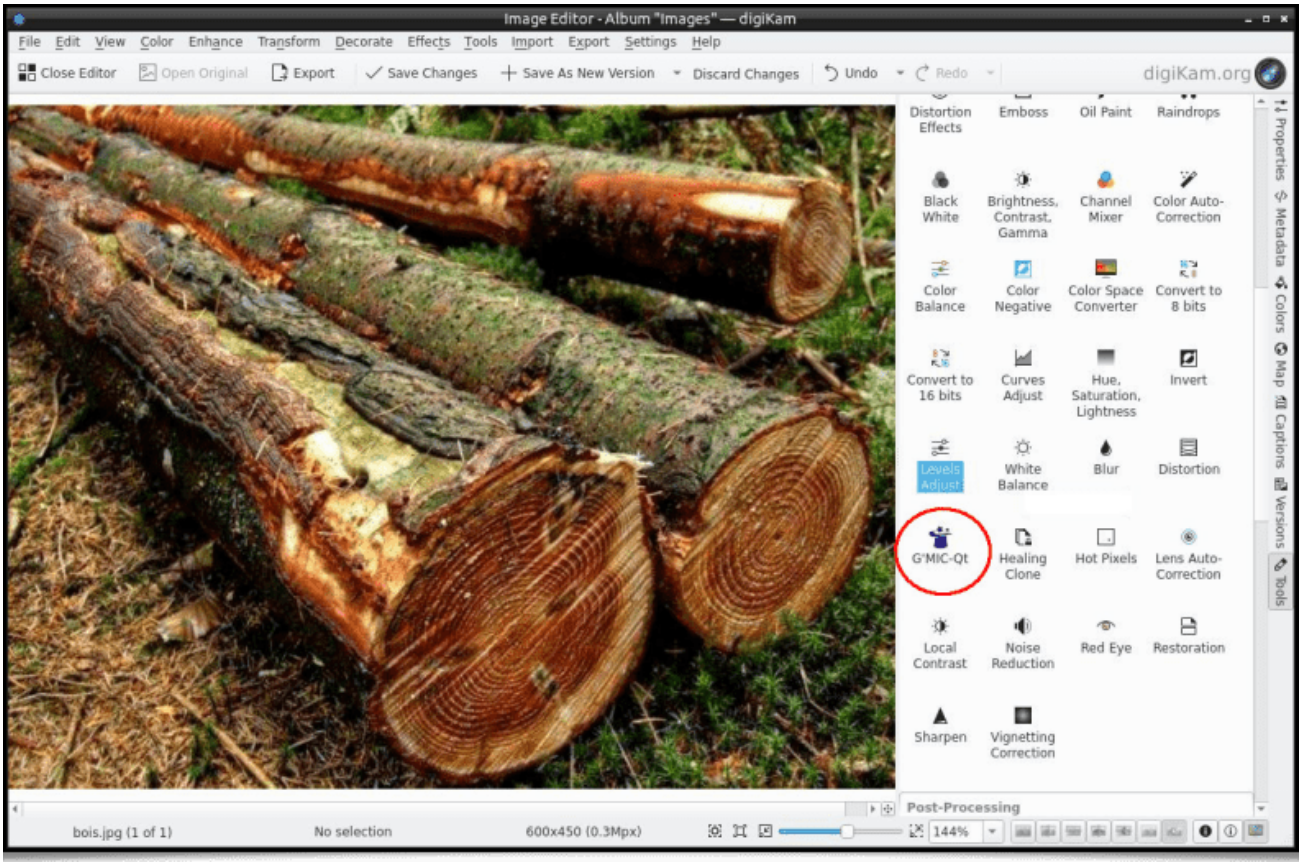

Fig. 5.1.2. Le greffon G'MIC-Qt est disponible directement depuis Digikam, logiciel libre de gestion de photographies.

5.2. Amélioration de la bibliothèque standard `stdgmic`

La bibliothèque standard de *G'MIC* (`stdgmic`) contient l'ensemble des commandes non natives, écrites directement en langage *G'MIC*, et distribuées par défaut avec le cadriciel. En pratique, la grand majorité des commandes existantes rentrent dans ce cadre. En plus des nouvelles commandes déjà décrites plus haut, notons les ajouts et améliorations suivantes dans la `stdgmic` :

- La bibliothèque `nn_lib`, permettant l'**apprentissage de réseaux de neurones** simples, se dote de nouveaux modules (*Softmax Layer*, *Cross-Entropy Loss*, *Binary Cross-Entropy Loss*, *Conv3D*, *Maxpool3D* et *PatchUp/PatchDown 2D/3D*). Son développement avance doucement. Elle est déjà utilisée par le filtre **Repair / Denoise**, **déjà mentionné** dans notre précédente dépêche. Nous avons aussi implémenté quelques exemples « jouets » d'apprentissage statistique utilisant cette bibliothèque, comme par exemple l'apprentissage d'une fonction *(x,y)* → *(R,G,B)* représentant une image. L'idée ici est d'apprendre à un réseau de neurones à reproduire une image couleur, en lui fournissant comme données d'apprentissage les coordonnées *(x,y)* des points de l'image (en entrée) et leurs couleurs *(R,G,B)* correspondantes (en sortie). La figure ci-dessous montre l'image globale reconstruite par le réseau au fur et à mesure des itérations d'apprentissage :

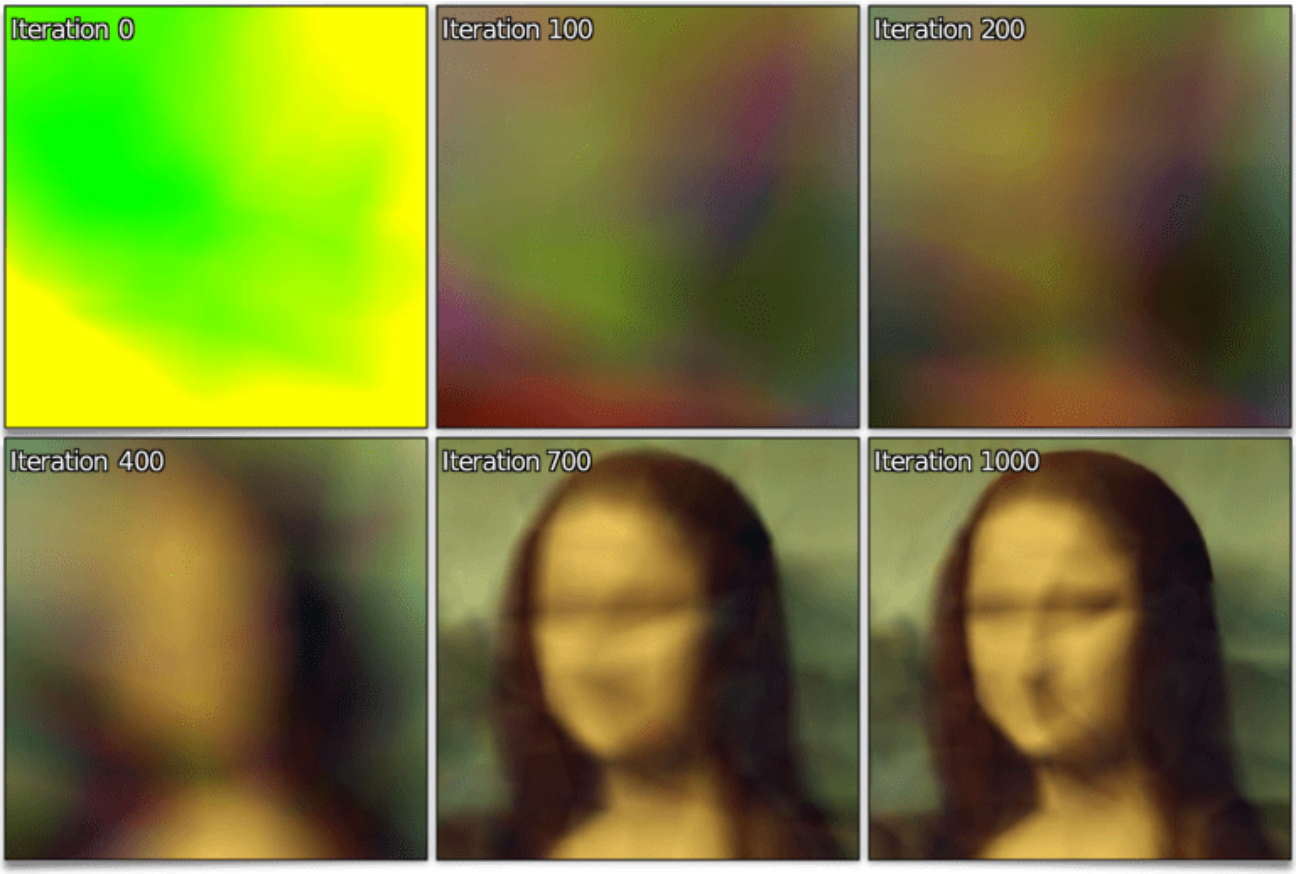

Fig. 5.2.1. Apprentissage par réseaux de neurones d'une fonction (x,y) → (R,G,B) représentant une image. Les différentes itérations du processus d'apprentissage sont affichées.

La séquence d'apprentissage complète est visible dans la vidéo ci-dessous :

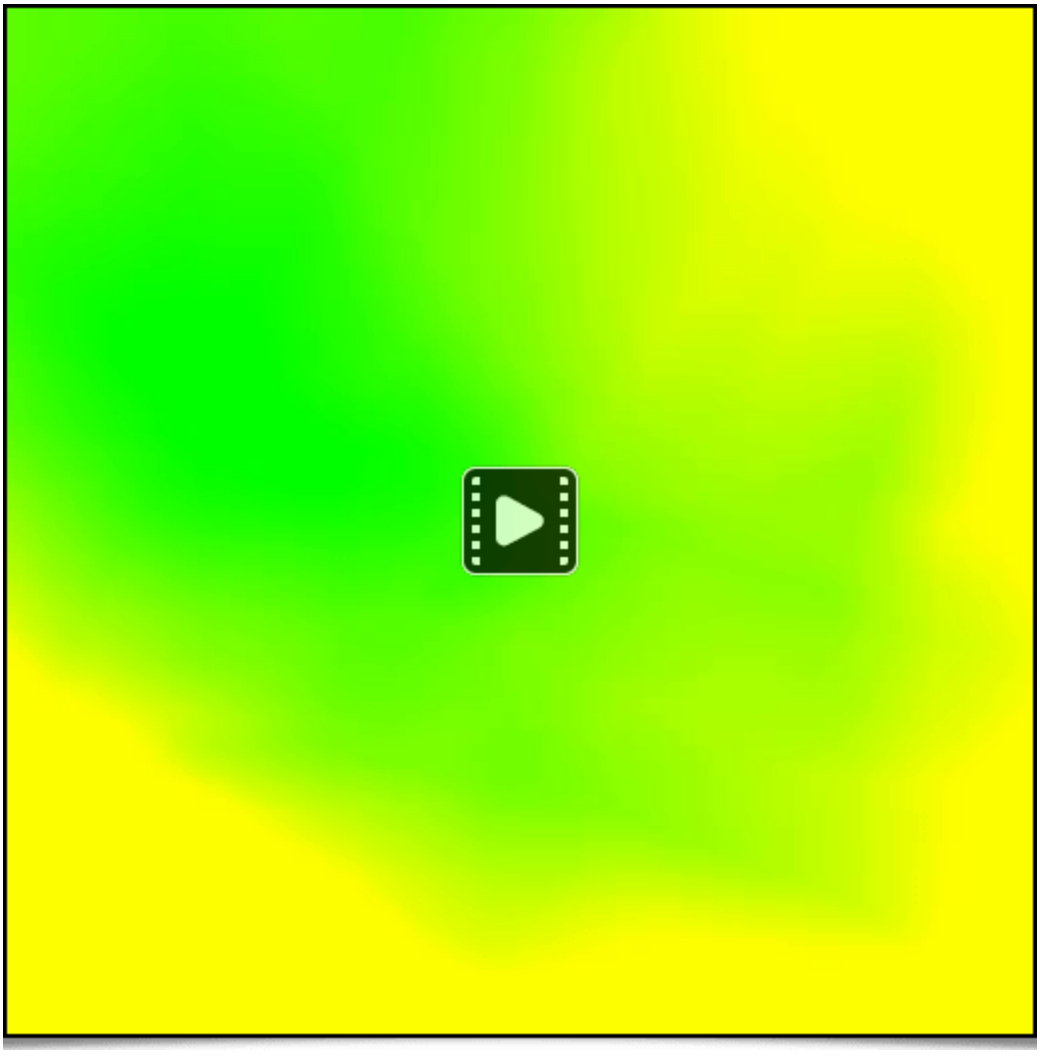

Fig. 5.2.2. Séquence des itérations d'apprentissage du réseau de neurones, pour l'apprentissage d'une fonction (x,y) → (R,G,B) représentant une image.

Nous avons également des exemples fonctionnels de la `nn_lib` pour classifier automatiquement des images simples (provenant entre autres des jeux de données ***MNIST*** et ***Fashion MNIST***). *G'MIC* peut donc potentiellement détecter le contenu de certaines images, comme illustré ci dessous avec la classification de chiffres manuscrits (nous avons en stock une méthode similaire pour la détection de visages humains).

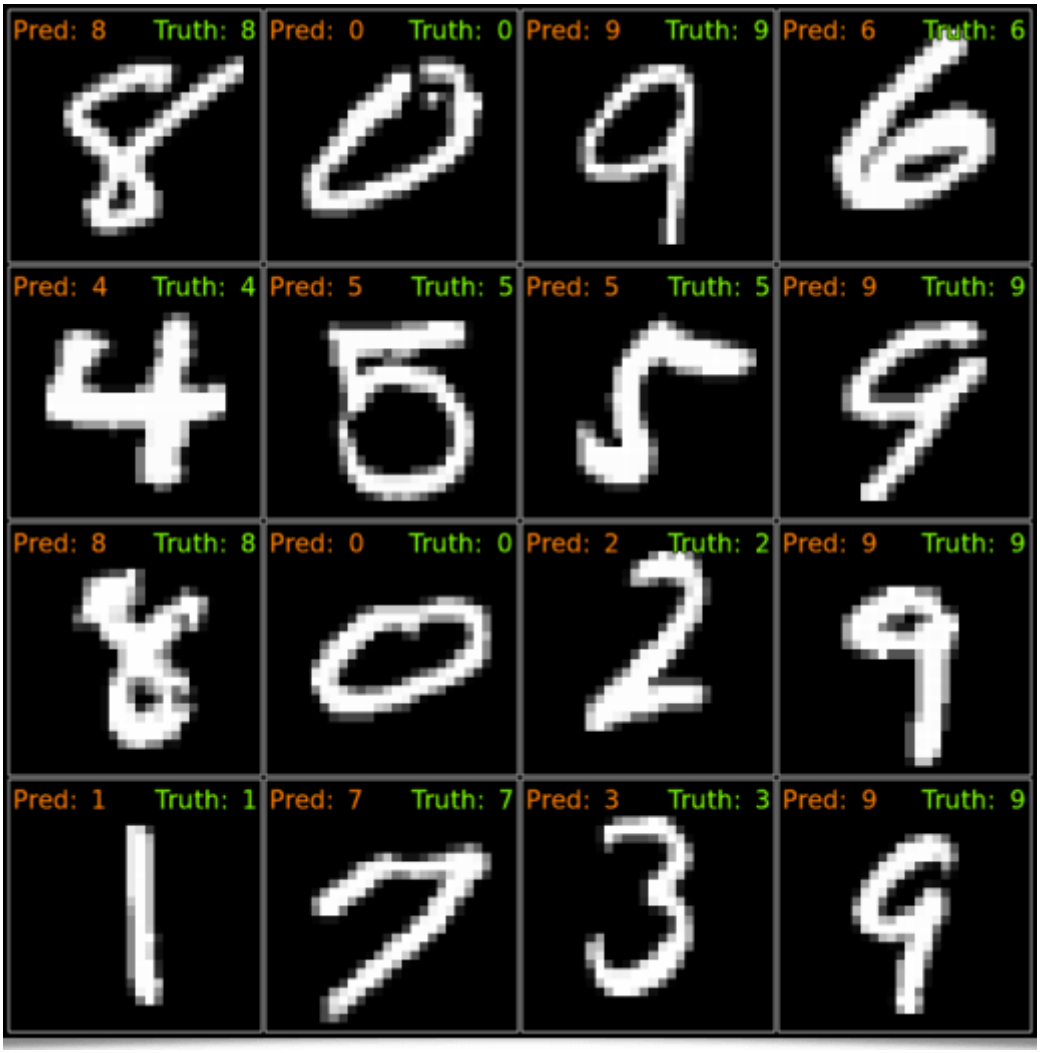

Fig. 5.2.3. Classification automatique d'images de chiffres manuscrits (base de données _MNIST) par réseau de neurones, via l'utilisation de la bibliothèque `nn_lib` de G'MIC_.

Un début de documentation sur l'utilisation de cette bibliothèque d'apprentissage statistique en langage *G'MIC* est disponible sur **notre forum de discussion**.

- Autre fonctionnalité de la `stdgmic` : la commande `match_icp` implémente l'algorithme appelé « **Iterative Closest Point** [W] » (*ICP*), qui met en correspondance deux ensembles de vecteurs à *N* dimensions. Cette commande peut servir à déterminer la transformation géométrique rigide (rotation + translation) entre deux *frames* d'une animation, et cela, même en présence de bruit. L'animation ci-dessous illustre ce processus, avec deux transformations rigides estimées par *ICP*, pour recaler respectivement les silhouettes d'étoile et de coeur.

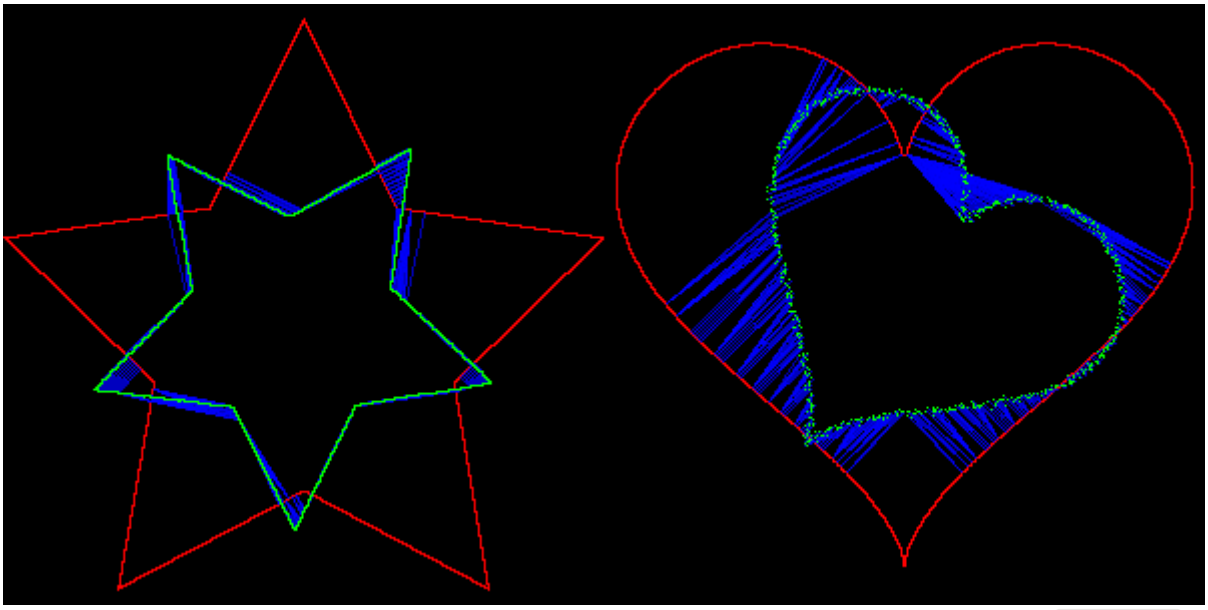

Fig. 5.2.4. Recalage de silhouettes par algorithme d'« Iterative Closest Point », via la commande `match_icp`.

- Signalons aussi les nouvelles commandes `img2patches` et `patches2img` : elles permettent respectivement la décomposition d'une image en un empilement volumique d'imagettes (« *patchs* ») et sa récomposition, éventuellement en prenant en compte des *patchs* qui se superposent. Par exemple, la commande :

```
$ gmic butterfly.jpg img2patches 64
```

va transformer l'image d'entrée (à gauche ci-dessous) en un empilement volumique de *patchs* (image de droite) :

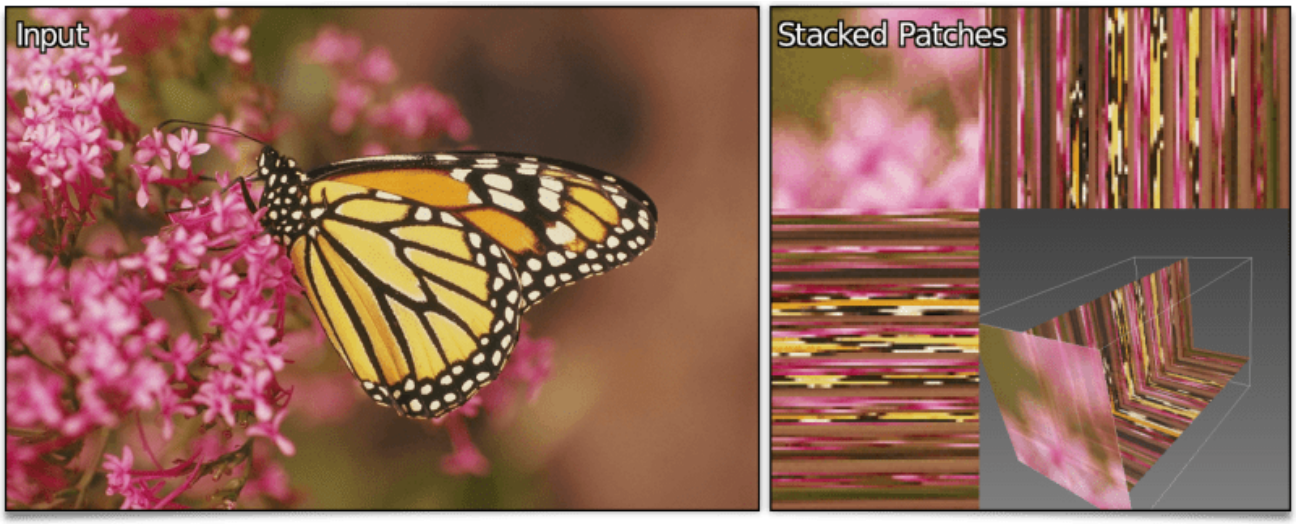

Fig. 5.2.5. Transformation d'une image couleur en un empilement volumique de _patchs, par la commande `img2patches`._

On peut ensuite traiter cet ensemble de *patchs*, par exemple en les triant dans l'ordre croissant de leur variance (donc du niveau de détails qu'ils contiennent), avant de recomposer l'image. Comme avec la commande suivante :

```
$ gmic sample butterfly W,H:=w,h img2patches 64,0,3 split z sort_list +,iv append z patches2img '$W,$H
```

Ceci produit l'image ci-dessous, où les *patchs* sont de plus en plus détaillés au fur et à mesure que l'on descend dans l'image.

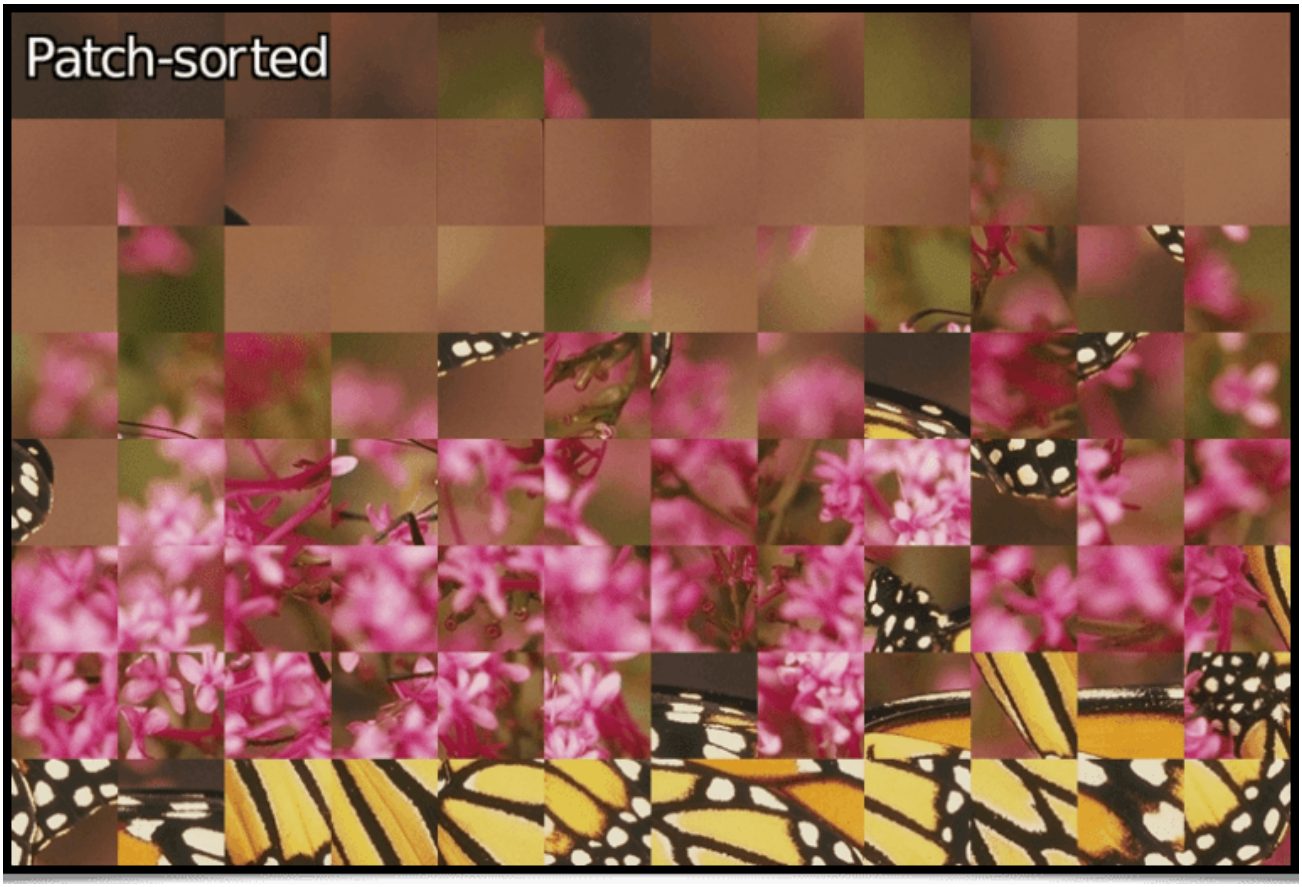

Fig. 5.2.6. Tri des patchs d'une image par ordre de détail croissant.

- La nouvelle commande `line aa`, implémente l'algorithme de **Xiaolin Wu**[W] pour le tracé de segments ***anti-aliasées***[W], c'est-à-dire qu'elle essaye de minimiser l'effet de crénelage qui apparaît classiquement lors du tracé de primitives dans des images discrètes.

Fig. 5.2.7. Comparaison des rendus de tracés de segments, entre l'algorithme classique de Brensenham et l'algorithme de Xiaolin Wu implémenté par la commande `line_aa`.

- Pour achever cette section sur la bibliothèque standard de *G'MIC*, mentionnons l'arrivée des commandes `ssim` (calcul de la mesure « ***Structural Similarity***[W] » entre deux images), `opening`, `opening_circ`, `closing`, `closing_circ` (ouverture et fermeture morphologiques avec élément structurant carré ou circulaire), `betti` (calcul des **nombres de *Betti***[W], invariants topologiques de formes discrétisées en 2D ou 3D) et d'un nouveau mode de fusion de calque `shapeprevalent` pour la commande `blend`. Comme vous le voyez, il y a toujours de nouvelles choses à explorer ☺ !

5.3. Informations diverses liées au projet

Pour terminer cette longue dépêche, voici en vrac, quelques informations générales sur le projet *G'MIC*.

- D'abord, un rappel de l'existence de greffons ***OpenFX***[W] embarquant les fonctionnalités de *G'MIC*, permettant donc d'appliquer la plupart de nos filtres dans les logiciels d'édition vidéo qui implémentent cette *API* (tels que ***Natron***[W] ou ***Adobe After Effects***[W]). Pour en savoir plus, voir **le *post* dédié** de l'auteur de ces greffons, **Tobias Fleischer**.

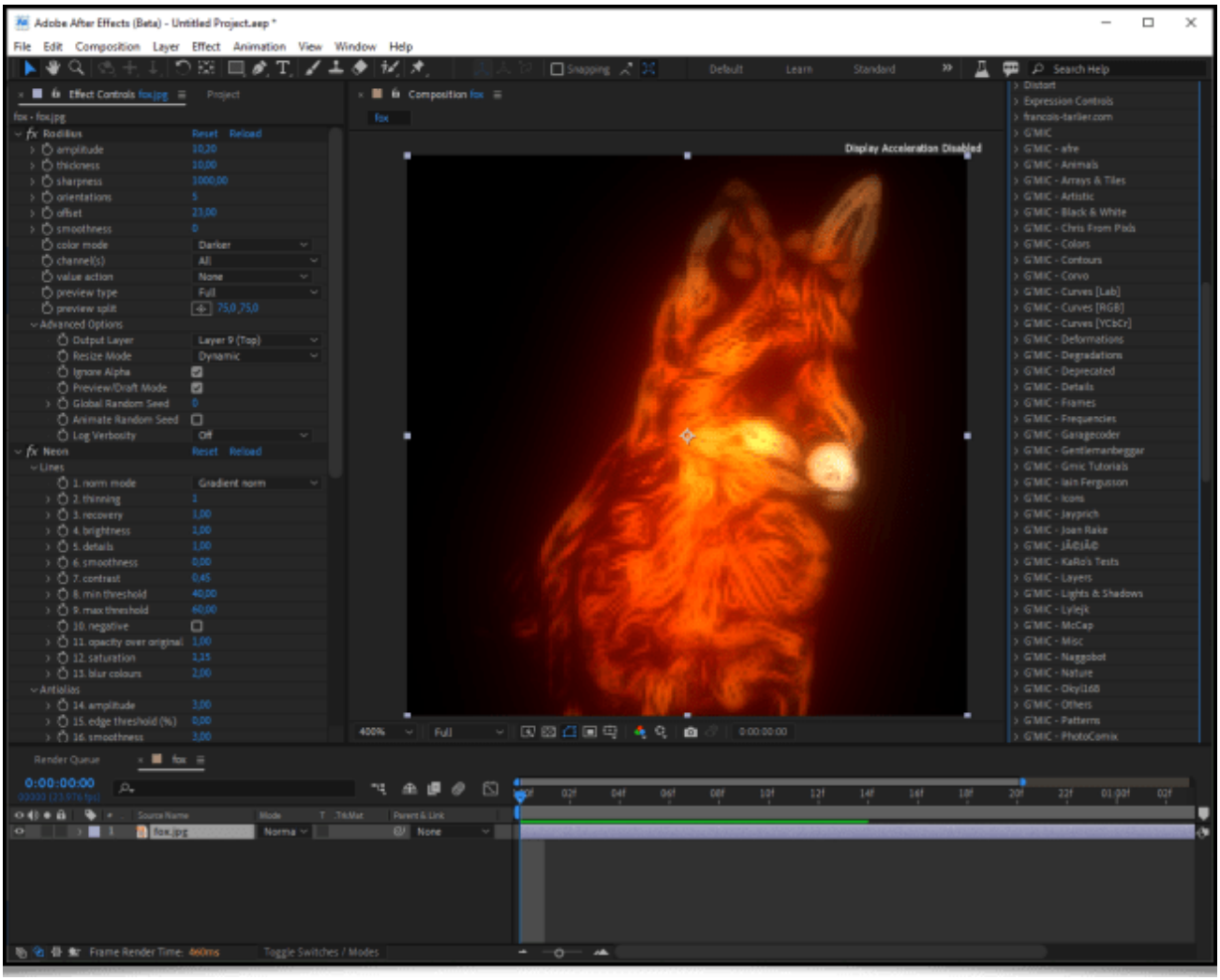

Fig. 5.3.1. Un des greffons OpenFX de G'MIC en action, dans le logiciel Adobe After Effects.

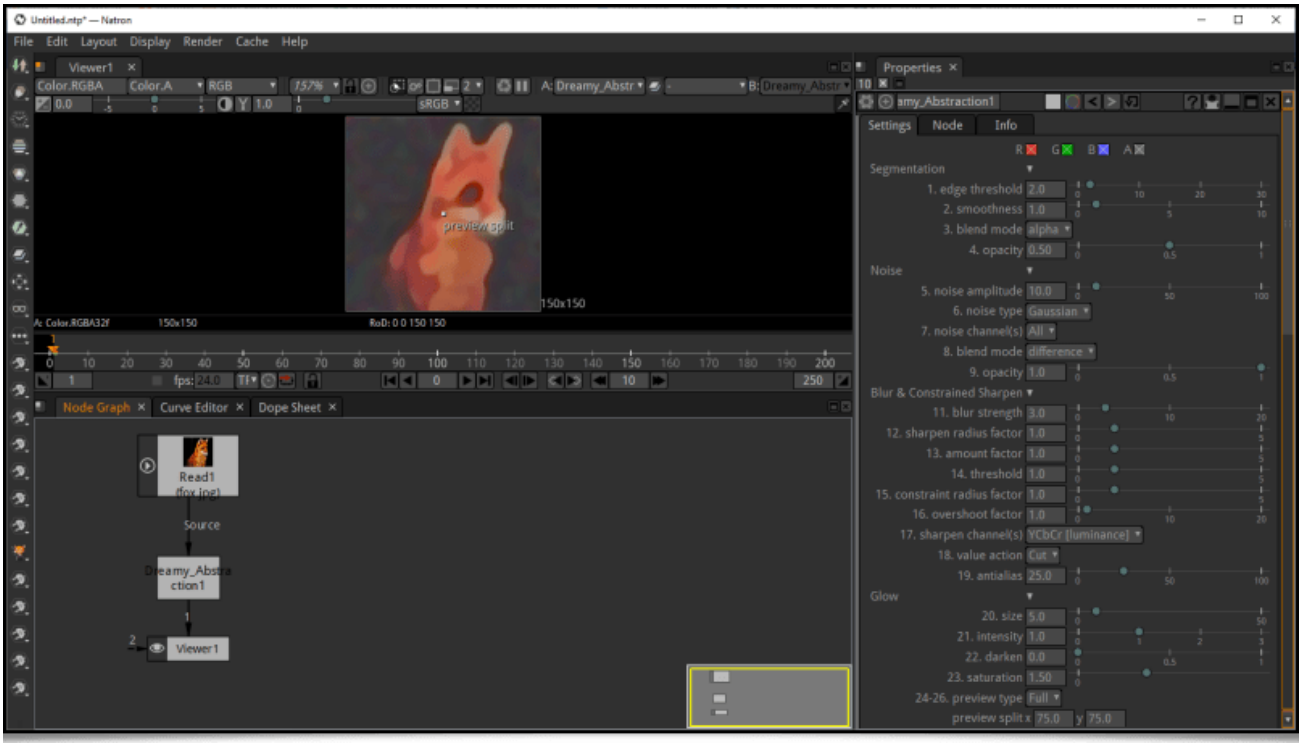

Fig. 5.3.2. Un des greffons OpenFX de G'MIC en action, dans le logiciel Natron.

- Notre algorithme d'**éclairage automatique de dessins colorisés en aplats**, mentionné lors d'une dépêche précédente (filtre **Illuminate 2D Shape**) a fait l'objet d'une publication en fin d'année 2022, à la conférence ***IEEE International Conference on Image Processing*** à Bordeaux. Cette publication, intitulée « ***Automatic Illumination of Flat-Colored Drawings by 3D Augmentation of 2D Silhouettes*** » détaille l'ensemble de l'algorithme implémenté dans ce filtre. Ce filtre est apprécié des illustrateurs, qui peuvent l'utiliser pour donner rapidement du relief à leurs dessins colorisés en aplats, comme le montre la vidéo suivante :

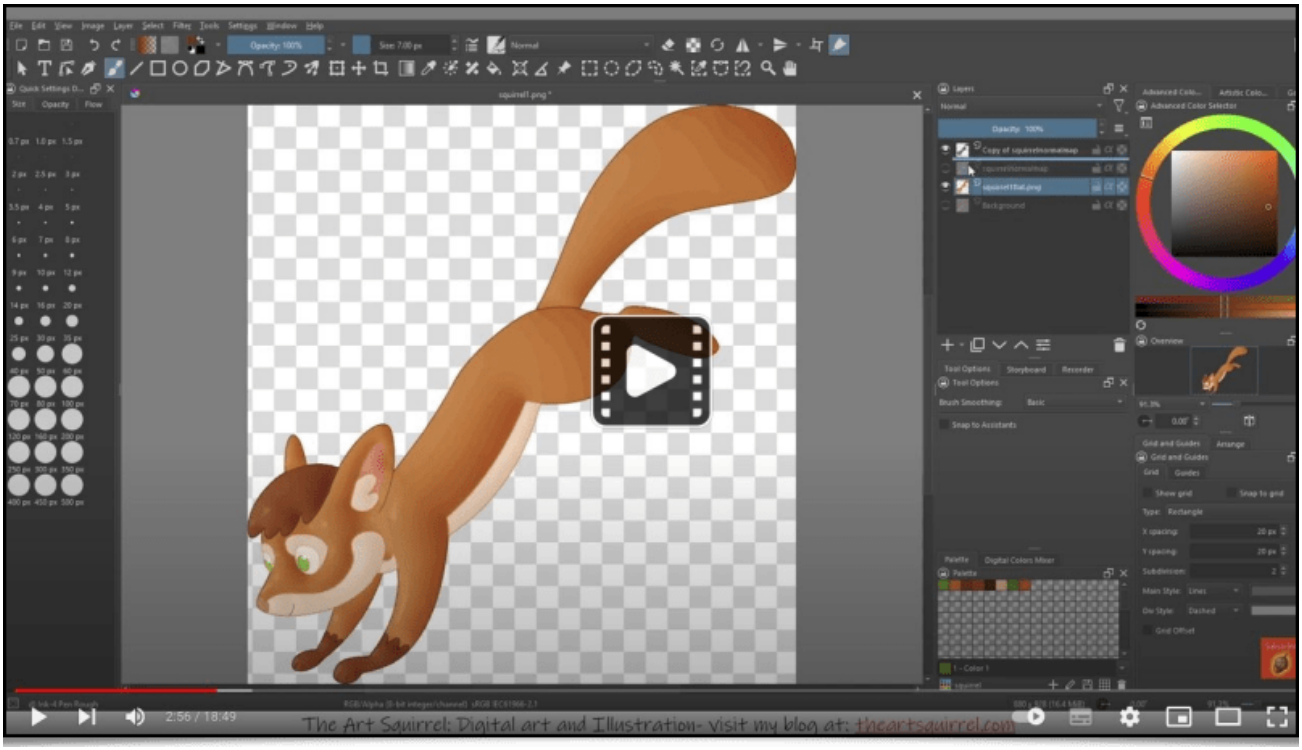

*Fig. 5.3.3. Tutoriel vidéo d'utilisation du filtre **Illuminate 2D Shape** pour l'éclairage automatique d'un dessin colorisé en aplats.*

Les plus curieux de détails techniques sur l'algorithme pourront visualiser la présentation suivante, donnée à la conférence *ICIP'2022* :

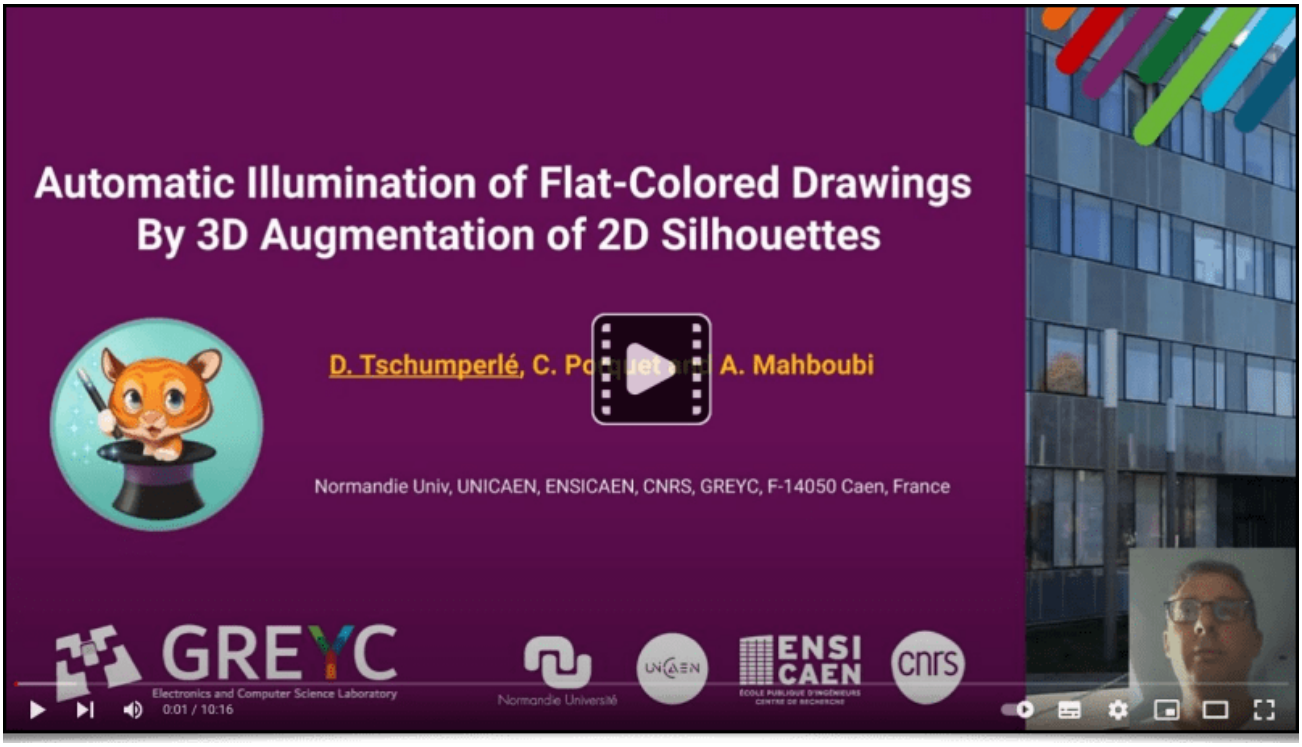

*Fig. 5.3.4. Vidéo d'explication de l'algorithme à la base du filtre **Illuminate 2D Shape**.*

- Notons que le langage de script de *G'MIC* est suffisamment flexible pour être utilisé, non seulement pour définir des filtres de traitement d'images, mais aussi pour élaborer des **démonstrateurs interactifs**. Au laboratoire *GREYC*, il nous a permis de développer deux bornes de démonstration autour du traitement d'images. Ces démonstrateurs sont exposés sur notre stand lors d'évènements grand-public (par exemple la **Fête de la Science**, ou le **Festival de l'Excellence Normande**).

Le premier de ces démonstrateurs est visible en cliquant sur l'image ci-dessous (présentation donnée par notre collègue **Loïc Simon**). Il illustre la problématique du « *transfert de style* » entre deux images. Il s'exécute sur une table tactile.

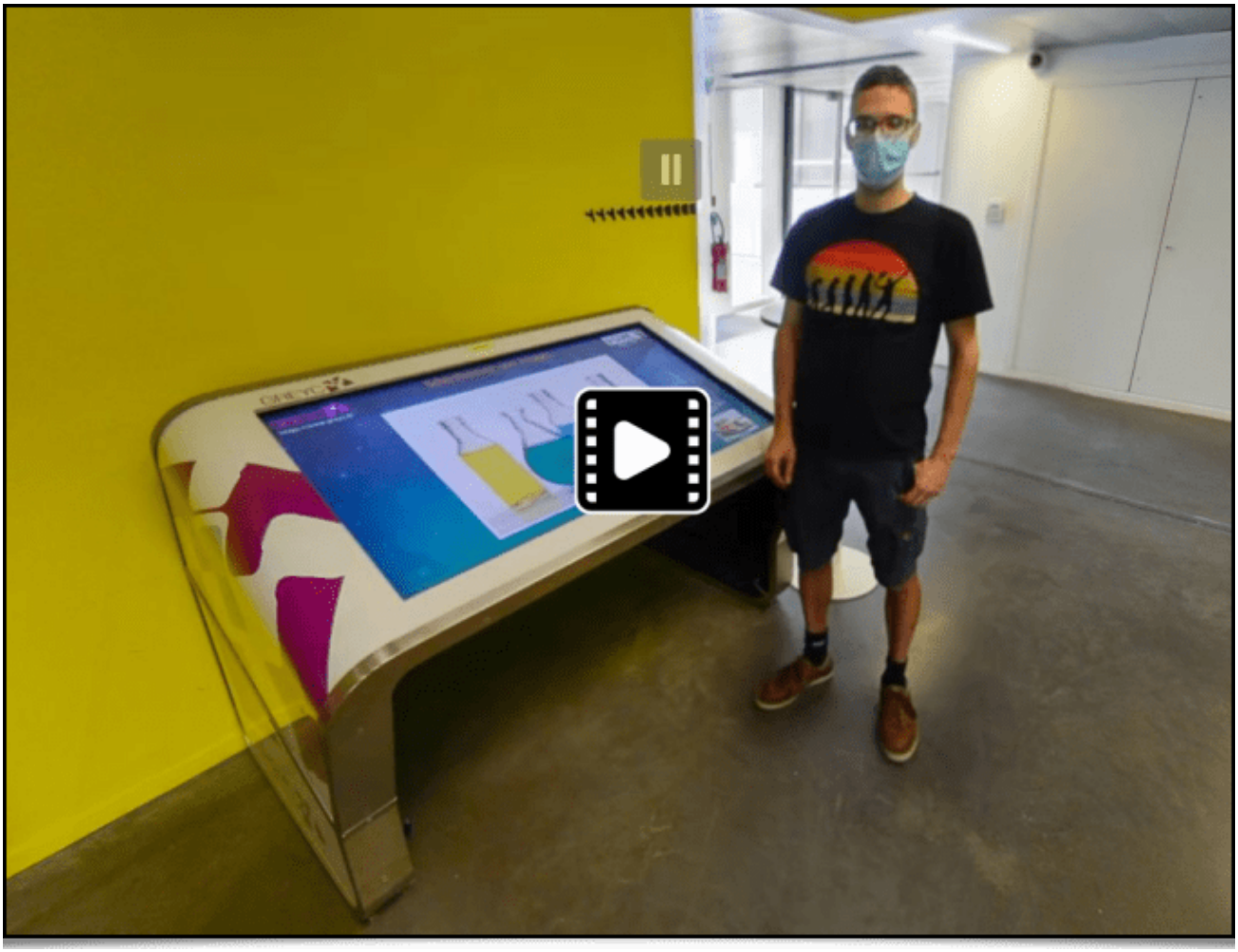

Fig. 5.3.5. Aperçu du démonstrateur G'MIC de transfert de style (cliquez sur l'image pour accéder à la démo à 360°).

Le deuxième démonstrateur permet de jouer avec un miroir déformant interactif, comme montré dans la vidéo ci-dessous :

Fig. 5.3.6. Démonstrateur interactif de déformation d'images, implémenté en langage G'MIC.

- À titre de projet personnel, j'ai commencé l'écriture d'un ***raytracer***[W] simple en langage *G'MIC*, pour tester les capacités du langage. Le but n'est pas forcément d'aller très loin (par manque de temps, car c'est très intéressant en pratique !), mais c'est un bon moyen de détecter les optimisations intéressantes qui pourraient être faites dans l'interpréteur *G'MIC* ultérieurement. Une animation d'un objet simple, générée par ce *raytracer* en cours de développemnt, est visible ci-dessous :

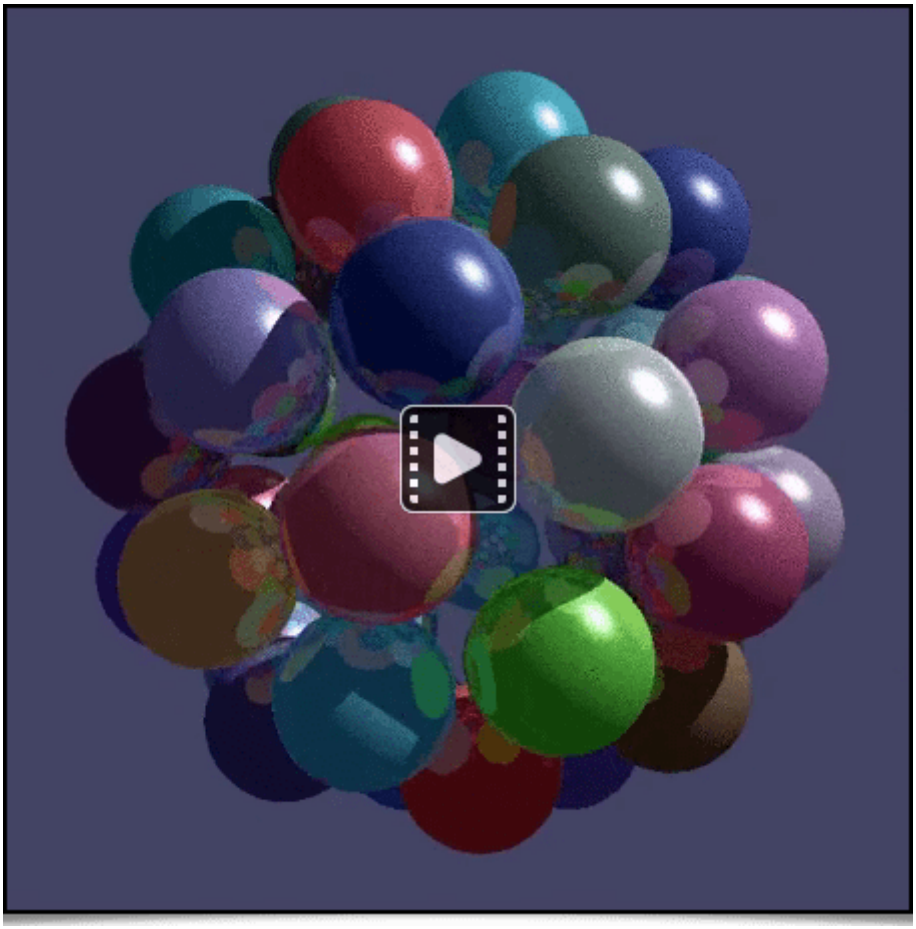

Fig. 5.3.7. Exemple de rendu obtenu par la technique du raytracing, implémentée en G'MIC (en cours de développement).

- Pour ceux qui veulent en savoir plus sur le fonctionnement du langage *G'MIC*, nous recommandons la lecture des **formidables pages de tutoriel** écrites par **Garry Osgood**, contributeur depuis plusieurs années à la documentation du projet *G'MIC*. En particulier, il a récemment écrit **une série d'articles sur la création d'arabesques**, que l'on ne peut que conseiller !

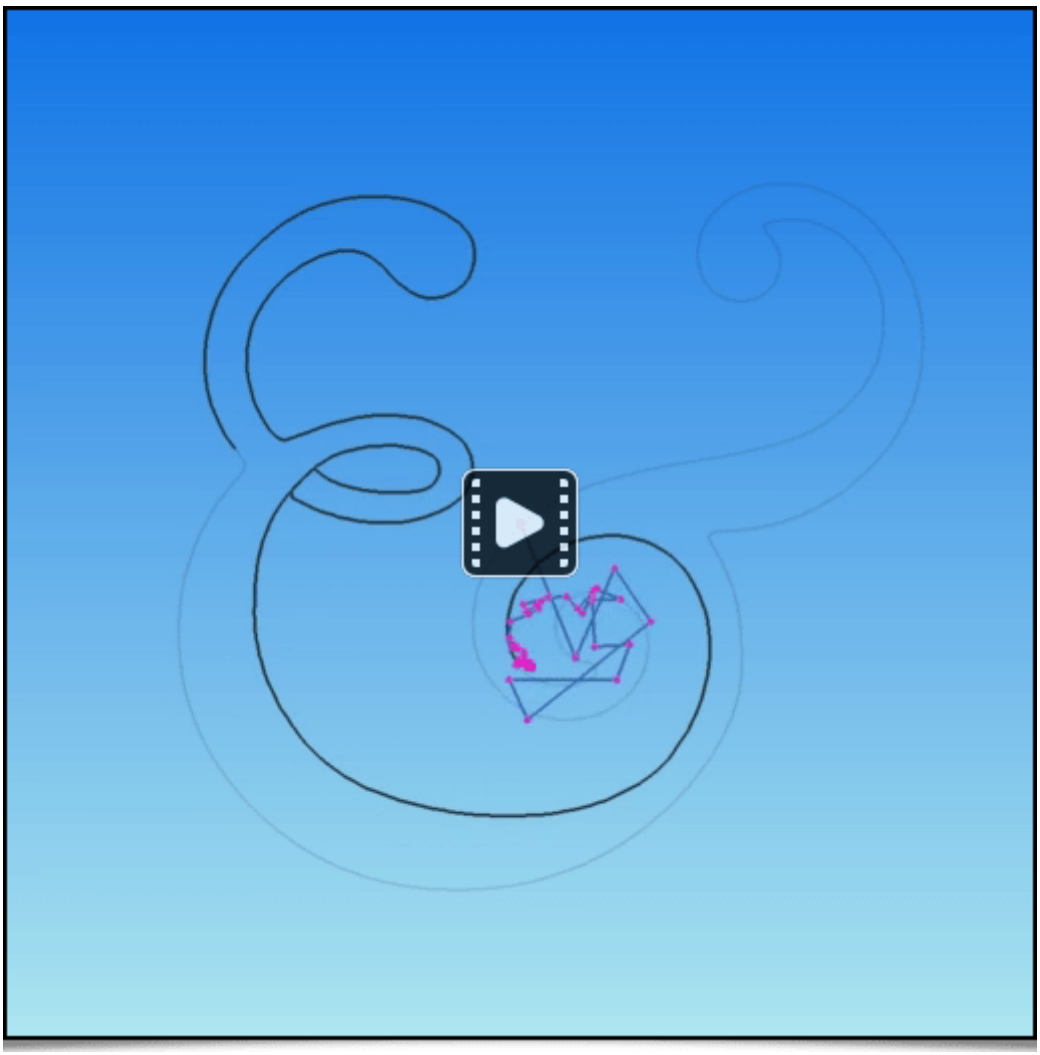

Fig. 5.3.8. Exemple de tracé de silhouette par la technique d'arabesque décrite dans le tutoriel de Garry Osgood.

- Notons qu'avec le langage *G'MIC*, il est aussi possible de créer des ***one-liners*** amusants, c'est-à-dire des commandes qui tiennent sur une ligne et qui génèrent des images ou des animations insolites. Les deux commandes *G'MIC* suivantes en sont de bons exemples :

One-liner N°1 : Génération d'une image couleur fixe.

```
$ gmic 500,500 repeat 10 { +noise_poissondisk[0] '{3+$>}' } rm[0] a z f '!z?(R=cut(norm(x-w/2,y-h/2)/2
```

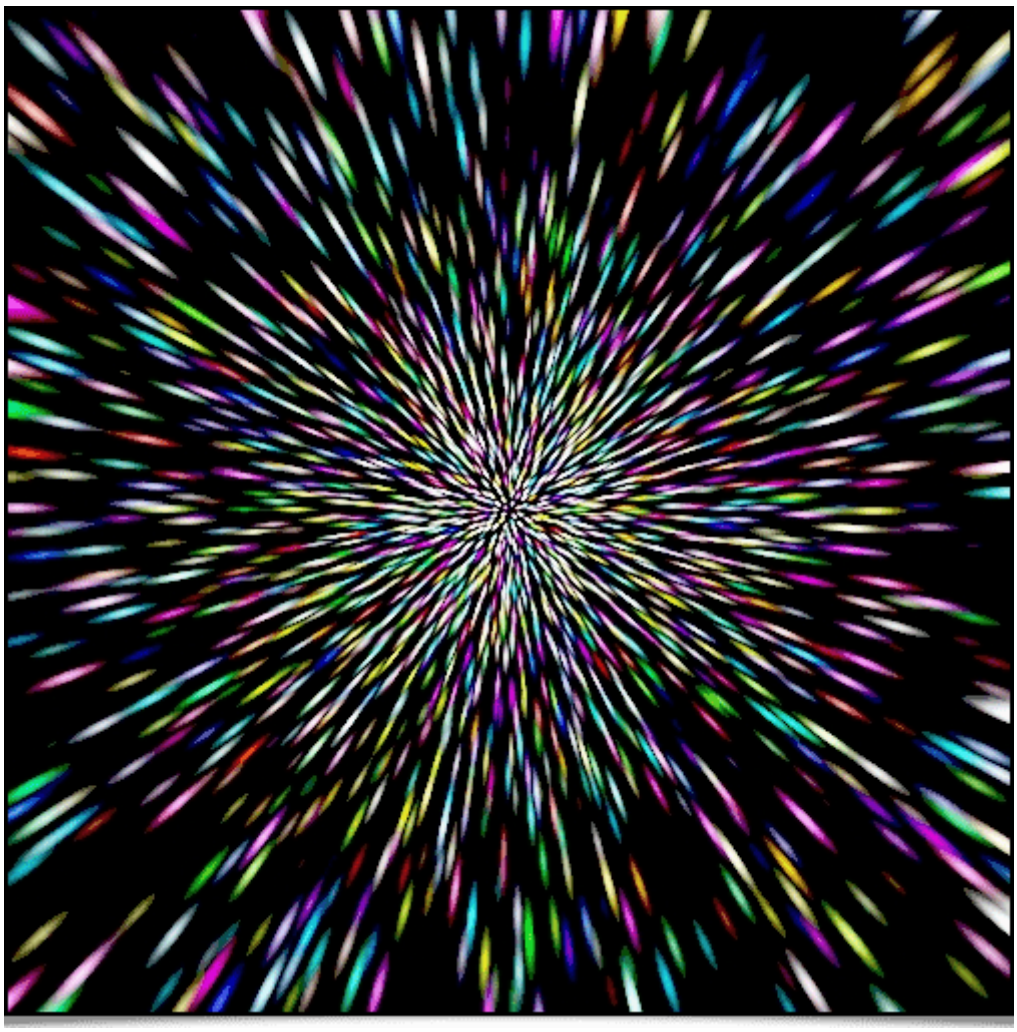

Fig.5.3.9. Résultat du one-liner N°1.

One-liner N°2 : Création d'une animation couleur, de type « peau de dinosaure ».

```
$ gmic 300,300x5 foreach { noise_poissondisk 40 +distance 1 label_fg.. 0 mul. -1 watershed.. . rm. g x
```

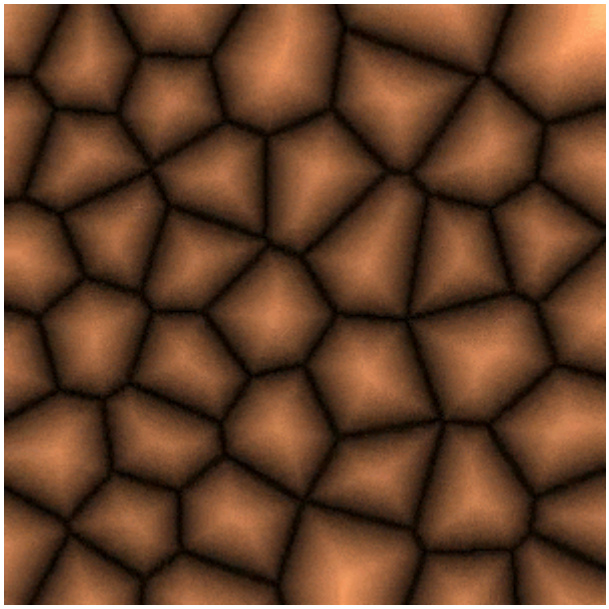

Fig.5.3.10. Résultat du one-liner N°2.

- Les deux images ci-dessous sont le résultat d'expérimentations en langage *G'MIC* de ***Reptorian***, contributeur de longue date, qui explore beaucoup les possibilités du langage pour l'art génératif.

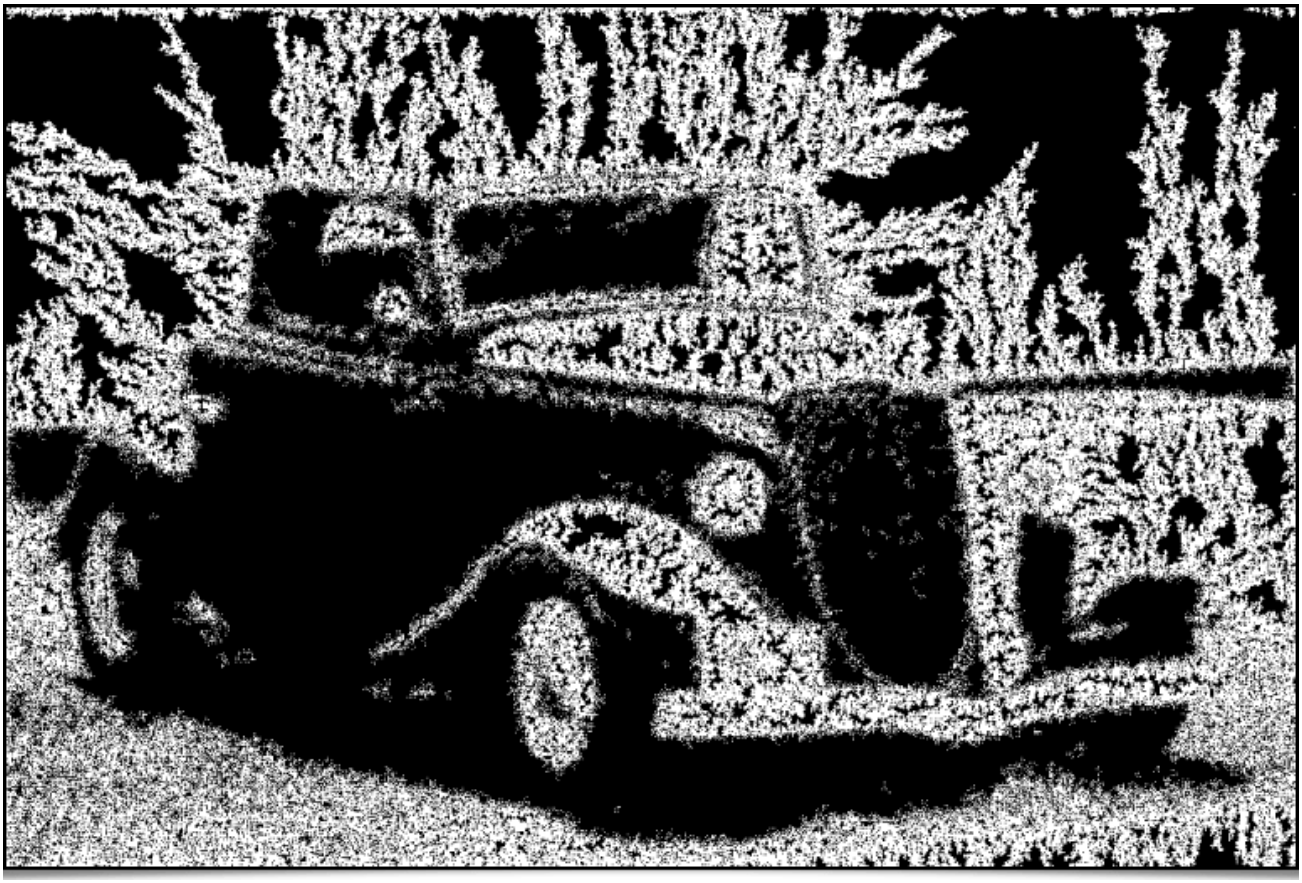

Fig.5.3.11. Variante de la technique de « Diffusion-limited aggregation[W] », guidée par la géométrie d'une image (par Reptorian).

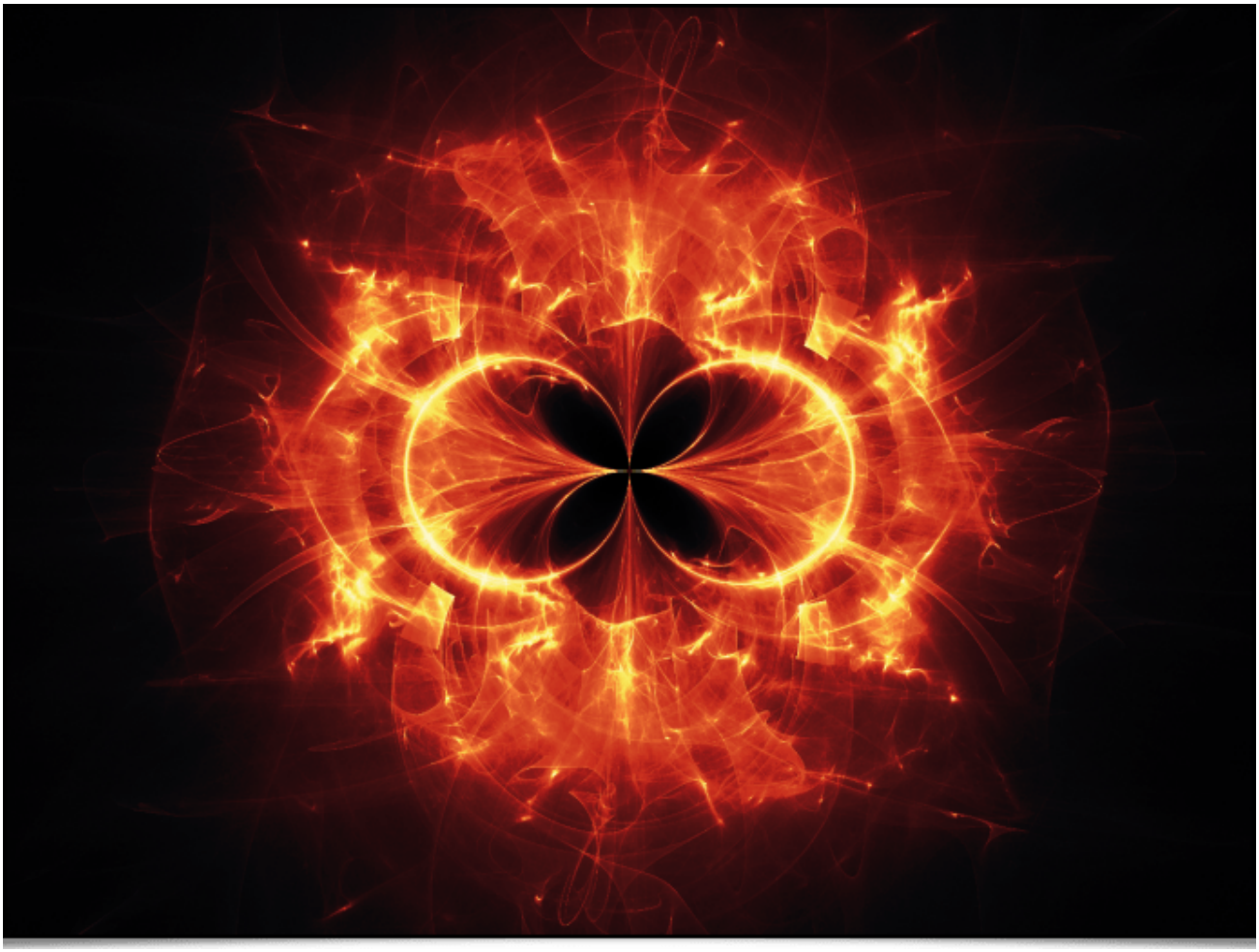

Fig.5.3.12. Génération de motif fractal (par Reptorian).

D'autres exemples sont visibles sur **son fil de discussion** (en anglais).

- Concernant la dimension « communication » du projet, il existe depuis plusieurs années un **compte *Twitter*** où nous postons régulièrement des informations sur l'évolution du projet, l'arrivée de nouvelles fonctionnalités, la sortie de nouvelles versions, etc. Dorénavant, nous avons également **un compte *Mastodon***, où nous postons les actualités autour du projet. N'hésitez pas à vous y abonner !

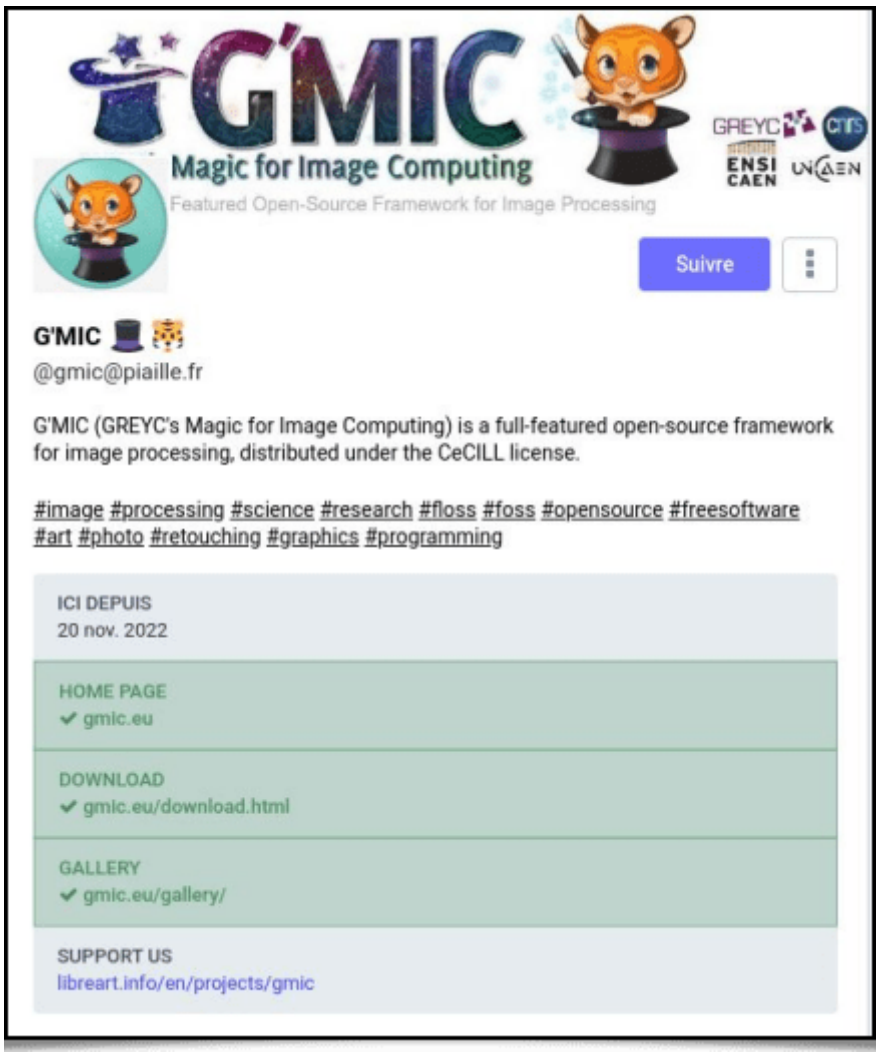

Fig. 5.3.13. Aperçu du compte Mastodon de G'MIC.

- Sur les réseaux sociaux, il arrive qu'on tombe sur des *posts* inattendus où des personnes montrent leur utilisation de *G'MIC*. Par exemple, cette série de *posts* récents met en jeu du traitement d'images astronomiques avec des filtres de *G'MIC*, pour débruiter ou esthétiser des images :

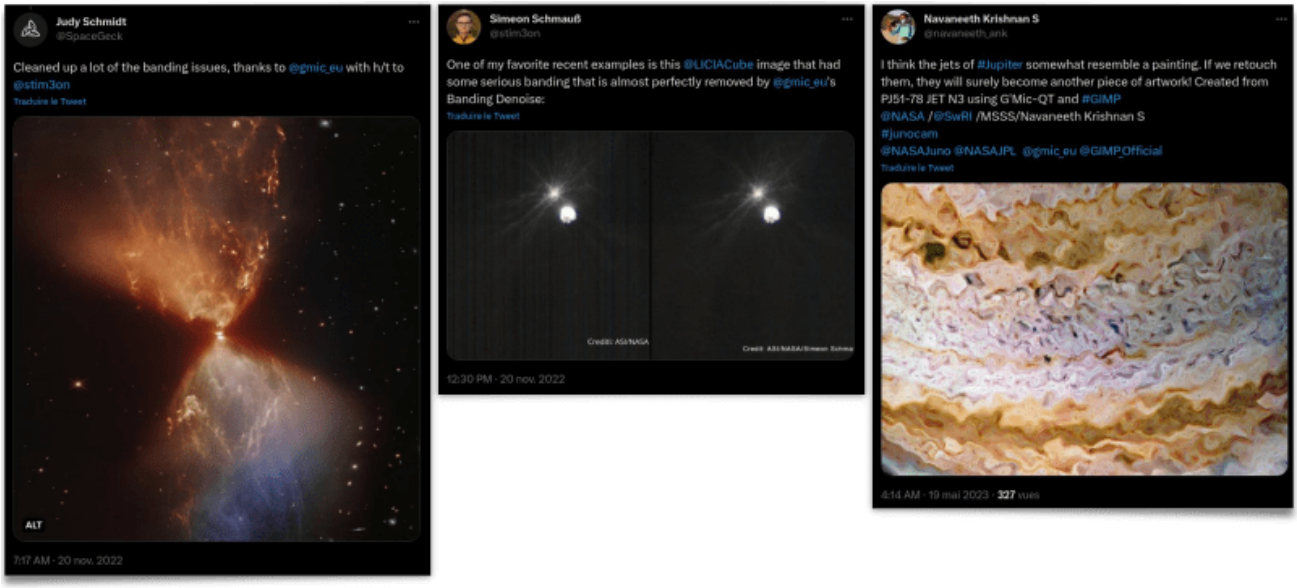

Fig. 5.3.14. Utilisation de G'MIC pour le traitement d'images astronomiques.

Ces *posts* sont visibles **ici**, **là**, **là**, ou **encore là**. Ces retours d'utilisateurs sont évidemment valorisants pour nous. Si vous êtes vous même utilisateurs (contents ☺) de *G'MIC*, n'hésitez surtout pas à partager vos réalisations ou vos retours d'expérience. Ca fait toujours plaisir !

- Enfin, mentionnons le fait que *G'MIC* a fait l'objet d'articles dans les numéros *301* et *302* du magazine anglais **LinuxFormat** (numéros de mai et juin 2023), rédigés par **Karsten Gunther**. Ils présentent les différentes possibilités de retouche photo apportées par le greffon *G'MIC-Qt*, d'une manière très pédagogique (comme nous essayons d'ailleurs de le faire avec nos dépêches sur ***Linuxfr*** !).


TUTORIALS Image editing

GIMP

Part Three

Don't miss next issue, subscribe on page 16!

Using enhanced layer features with G'MIC

Karsten Günther takes a look beyond GIMP's standard layer functions. Besides special GIMP operations, tasks provided by G'MIC loom into view.

OUR EXPERT

Fig. 5.3.15. Le magazine « Linux Format » propose une série d'articles sur l'utilisation de G'MIC-Qt, dans ses numéros de mai et juin 2023.

Voilà, ce qui conclut notre tour d'horizon des derniers développements et informations portant sur le projet *G'MIC*.

6. Conclusions & perspectives

Après **15 ans** de développement de *G'MIC* et **24 ans** de développement de ***CImg***, la bibliothèque *C++* qui lui sert de fondation, nous disposons aujourd'hui d'un cadriciel libre et ouvert pour le traitement des images numériques, qui est mature et qui a prouvé son utilité pour résoudre des problèmes variés de traitement d'images.
Le nombre de téléchargements n'a cessé de croître, depuis l'écriture des premières lignes de code (en 2008), ce qui montre que le projet est dynamique et attire un large éventail d'utilisateurs.

Ce dynamisme est il durable ? Nous avons bien sûr toujours des idées pour l'amélioration de ce cadriciel. Mais en même temps, avec les responsabilités professionnelles qui augmentent, le temps à consacrer à son développement diminue. La stratégie pour la suite sera donc de :

- Bien sélectionner les pistes d'amélioration sur lesquelles travailler.
- Tenter de trouver du temps de développement extérieur (soit bénévole, soit financé).

Sur le court-terme, nous sommes à la recherche de contributeurs pour :

- Avancer sur le développement du ***binding G'MIC*** pour ***Python***. Il faudrait le mettre à jour et consacrer suffisamment de temps à le tester en profondeur, pour rendre *G'MIC* utilisable sans *bugs*, directement depuis un programme écrit en *Python*. Le *binding* existant est fonctionnel et constitue déjà une bonne base de travail.
- Réussir à *packager G'MIC* pour ***macOS***[W]. Nous recevons en effet de nombreuses requêtes d'utilisateurs de *Mac* qui ne savent pas compiler et installer le greffon *G'MIC-Qt* pour *GIMP*. Nul doute qu'il soit possible d'améliorer cette situation, moyennant de l'aide extérieure.

Si vous pensez pouvoir contribuer sur l'un de ces deux points, **n'hésitez surtout pas à nous contacter !**

Enfin, la révolution induite par l'utilisation des réseaux de neurones dans le domaine du traitement des images numériques est primordiale. Sur ce point, *G'MIC* a du retard à rattraper. Nous nous sommes plutôt focalisés jus-

qu'à présent sur l'algorithmique « classique » du traitement d'images. Il faudrait pouvoir développer plus rapidement notre bibliothèque `nn_lib` pour parvenir à déployer des modèles neuronaux plus larges (quelques dizaines/centaines de millions de paramètres, ça serait déjà satisfaisant !), afin d'autoriser le traitement ou la synthèse d'images en utilisant des techniques plus avancées d'apprentissage statistique.

Comme vous le voyez, ce ne sont pas les idées qui manquent !

Pour finir, il faut rappeler que le développement de *G'MIC* n'aurait pas été possible sans les encouragements et le support du ***GREYC***, notre laboratoire, et de ses tutelles : l'**Institut INS2I du CNRS**, l'**Université de Caen Normandie**, et l'***ENSICAEN***. Un grand merci à eux pour leurs aides à différents niveaux lors de ces quinze dernières années de développement. Depuis quelque temps, on voit se mettre en place, dans le domaine de la recherche scientifique, des initiatives intéressantes de valorisation de la science ouverte et reproductible (**plan national pour la science ouverte**, plan de la **science ouverte du CNRS**…), et du logiciel libre (programme de valorisation **Open de CNRS Innovation**). Ce sont des signaux encourageants pour les chercheurs qui investissent souvent beaucoup de temps dans la création de communs numériques libres (logiciels, jeux de données, etc.), et qui ont parfois du mal à valoriser ces réalisations comme des contributions scientifiques d'importance.

Nous espérons que cette dépêche vous a plu. Rendez-vous dans quelques ~~mois~~ semestres, avec, on l'espère, encore de nombreuses nouveautés à partager avec vous !

Aller plus loin

- **Le projet G'MIC** (96 clics)
- **Fil Mastodon des nouvelles du projet** (13 clics)
- **Série d'articles G'MIC sur LinuxFr.org** (12 clics)

bravo david

Posté par **Christophe Turbout** le 31/05/23 à 09:59. Évalué à 8 (+7/-0).

> tout est dans le titre !
>
> Beau travail !

Re: bravo david

Posté par **Franckito** le 04/06/23 à 10:54. Évalué à 1 (+1/-0).

> Je plussoie ! Très bonne bibliothèque et excellent article, merci !

Recherche, licence libre et communication

Posté par **lagou** le 31/05/23 à 10:45. Évalué à 8 (+7/-0).

> Bonjour,
>
> Je ne suis malheureusement pas capable d'apprécier le travail vu que je ne retouche pas de photos, ni de près ni de loin. Par contre, j'aimerais bien connaître, du point de vue de la recherche, l'apport de l'usage d'une licence libre.
> Outre la reproductibilité permise qui est inhérente à la science, quels avantages (inconvénients ?) identifiez-vous à la publication du code sous licence libre, à sa mise à disposition sous forme de bibliothèque facilement ajoutable à tout type de logiciels (jusqu'aux plus professionnels) ? Est-ce que l'effet de mutualisation de l'effort de développement généralement associé à la publication sous licence libre peut être constaté ? Quelle est la part inédite propre à la recherche qui se trouve dans G'MIC ?