Katsushika Hokusai

G'MIC 2.7 - Process Your Images with Style!

David Tschumperlé — September 6th, 2019

The IMAGE team at the GREYC research laboratory is pleased to announce the release of version **2.7** of **G'MIC** (*GREYC's Magic for Image Computing*), its free, generic, extensible, and probably a little magical, framework for digital image processing.

The previous PIXLS.US article on this open-source framework was published a year ago, in August 2018. This new release is therefore a good opportunity to summarize the main features and milestones of the project's life over the past twelve months. Fasten your seat belts, the road is long and full of surprises!

Useful links:

- The G'MIC Project
- G'MIC Twitter Feed
- G'MIC Forum on PIXLS.US

G'MIC in 300 words

G'MIC is a piece of software that has been developed for more than 10 years now, mainly in C++, by two members of the IMAGE team of the GREYC lab: Sébastien Fourey and David Tschumperlé. It is distributed under the terms of the CeCILL free-software license. GREYC is a French public research laboratory located in Caen, specialized in digital sciences, under the head of three academic institutions: CNRS, University of Caen, and ENSICAEN.

The IMAGE team, one of the seven teams in the laboratory, is composed of researchers, professors, Ph.D. students and engineers, all specialized in the fields of algorithmics and mathematics of image processing.

Fig.1.1: G'MIC project logo, and its mascot "Gmicky" (designed by David Revoy).

G'MIC is cross-platform (*GNU/Linux*, *MacOS*, *Windows*, ...). It provides various user interfaces for manipulating *generic* image data, i.e. 2D or 3D hyperspectral images or sequences of images with floating-point values (which indeed includes "usual" color images). Around a thousand different processing functions are already available. However, arbitrarily many features can be added thanks to an integrated scripting language.

The most commonly used G'MIC interfaces are: the `gmic` command, that can be accessed from the command line (which is an essential complement to ImageMagick or GraphicsMagick), the *G'MIC Online* Web service, but above all, the plug-in G'MIC-Qt, available for the well-known image editing software GIMP, Krita, and Paint.net. It provides more than 500 different filters to apply on images.

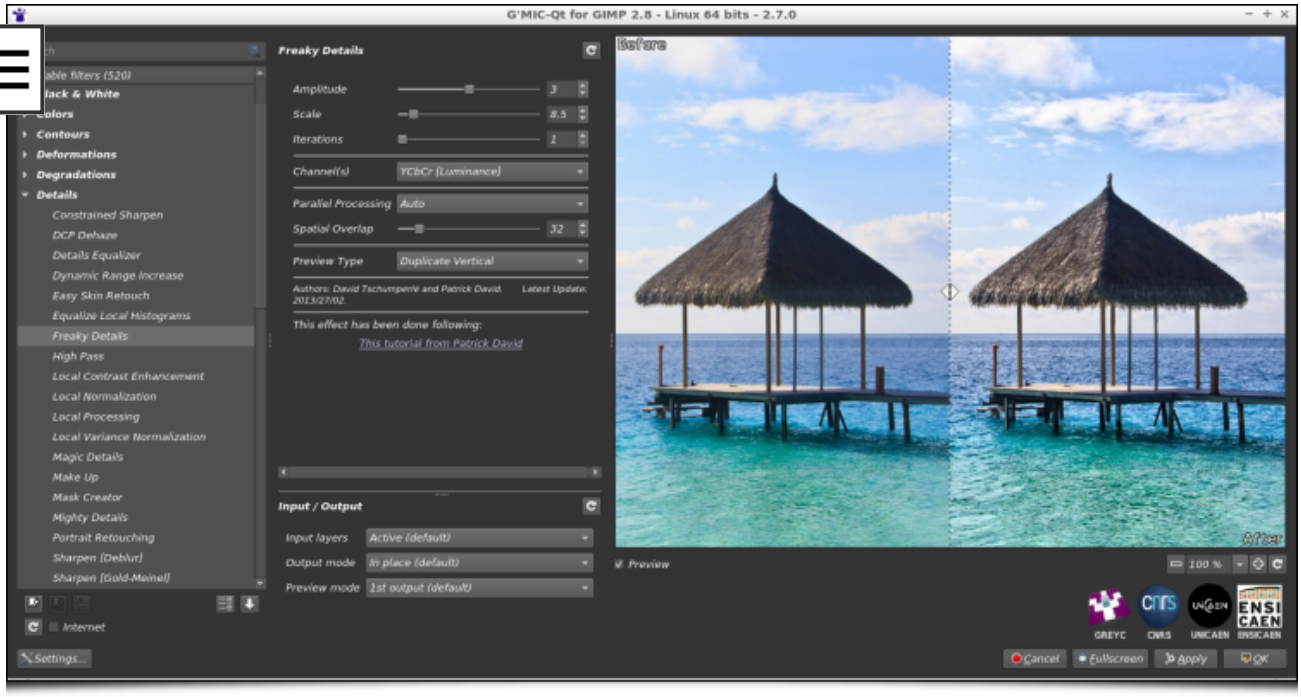


Fig.1.2: The G'MIC-Qt plug-in, here in version **2.7**, *is at the moment the most downloaded user interface of the G'MIC project.*

Thanks to its extensible architecture, G'MIC is regularly enhanced with new image processing algorithms, and it is these latest additions that will be discussed in the following sections.

2. Add style to your images!

G'MIC has recently implemented a neat filter for **style transfer** between two images, available from the G'MIC-Qt plug-in under the "**Artistic / Stylize**" entry. The concept of style transfer is quite simple: we try to transform an image (typically a *photograph*) by transferring the style of another image to it (for example a *painting*).

Fig.2.1: Principle of style transfer between two images.

The implementation of such a style transfer method is relatively complex: The algorithm must be able to recompose the original photograph by "borrowing" pixels from the style image and

intelligently combining them, like a puzzle to be reconstructed, to best match the content of the data to be reproduced, in terms of contours, colors and textures. How easily this is done depends of course on the compatibility between the input image and the chosen style. In computer graphics, most existing implementations of style transfer methods are based on convolutional neural networks, more particularly generative adversarial networks (GANs).

G'MIC implements style transfer in a different way (without relying on neural networks, the scientific article detailing the algorithm is currently being written!). This method is parallelizable and can therefore benefit from all the processing units (cores) available on the user's computer. The computation time naturally depends on the input image resolution, and the accuracy of the desired reconstruction. On a standard 4-cores PC, it could take tens of seconds for low resolution images (e. g. 800x800), up to several minutes for larger pictures.

As one can imagine, it is a **very versatile** filter, since we can apply any style to any input image without hard constraints. Some famous paintings are available by default in the filter, in order to propose predefined styles to the user.

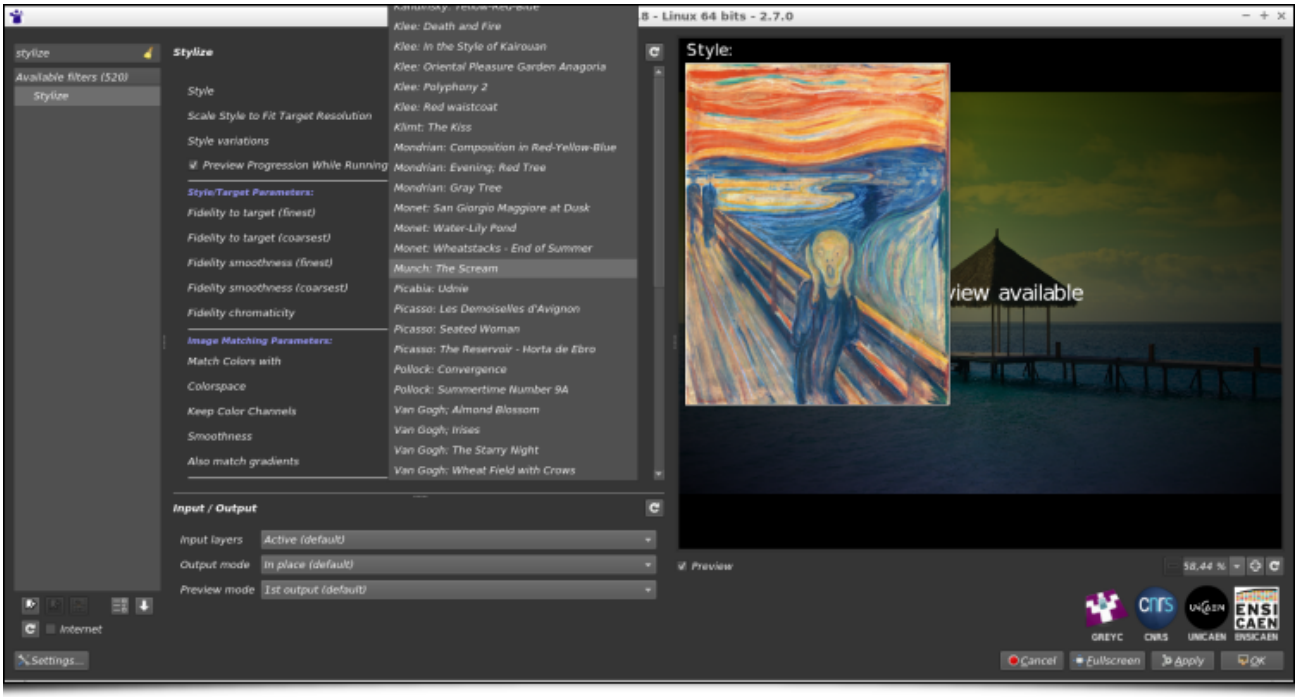


*Fig.2.2: "**Artistic / Stylize**" filter, as it appears in the G'MIC-Qt plug-in, with its many parameters that can be tuned !*

Let us be honest, it is not always easy to obtain satisfactory results from the first draft. It is generally necessary to choose

your starting images carefully, and to play with the many parameters available to refine the type of rendering generated by the algorithm. Nevertheless, the filter is sometimes able to generate quite interesting outcomes, such as those shown below (the original photo is visible at the top left, the style chosen at the top right, and the result of the style transfer at the bottom). Imagine how long it would take for a graphic designer to make these transformations “by hand”!

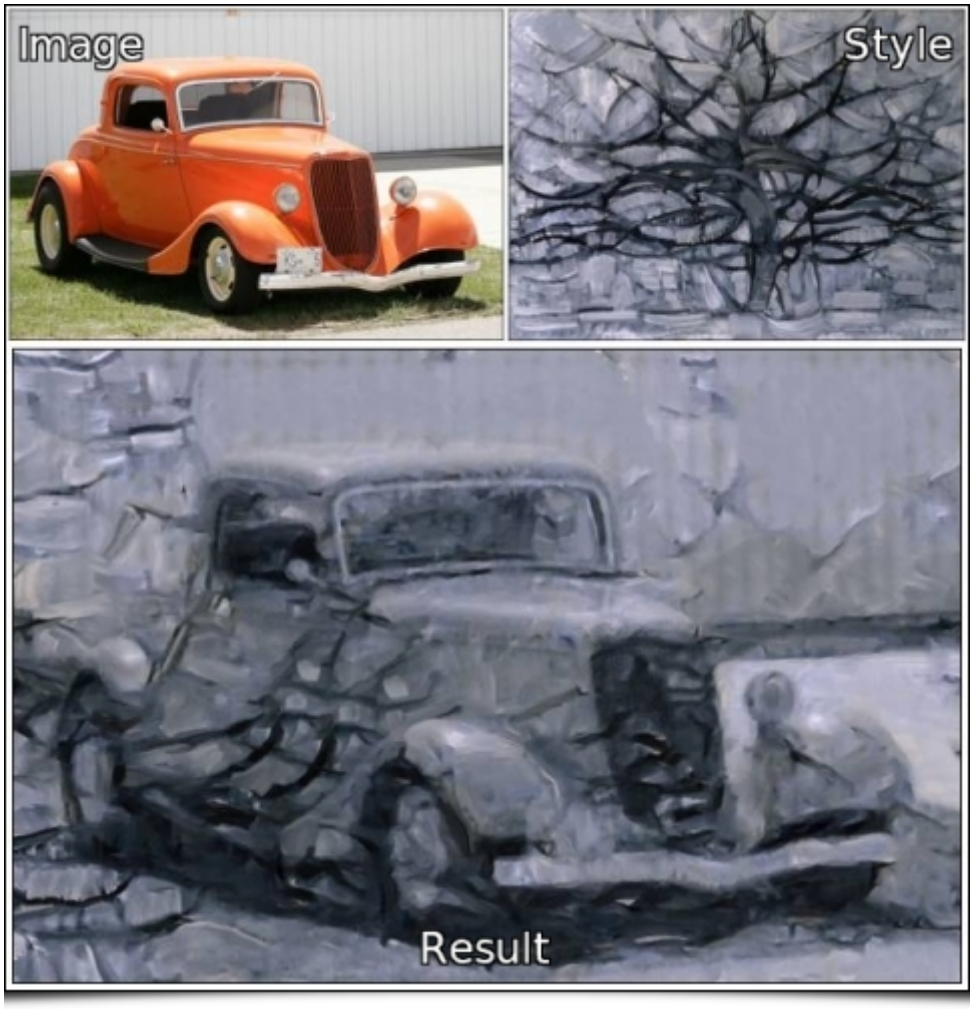


Fig.2.3: Stylization of a car from the painting “Gray Tree“ by Piet Mondrian.

Fig.2.4: Stylization of the same car from the painting "Gelb-Rot-Blau" by Vassily Kandinsky.

Fig.2.5: Stylization of the same car from the painting "The Great Wave off Kanagawa" of Hokusai.

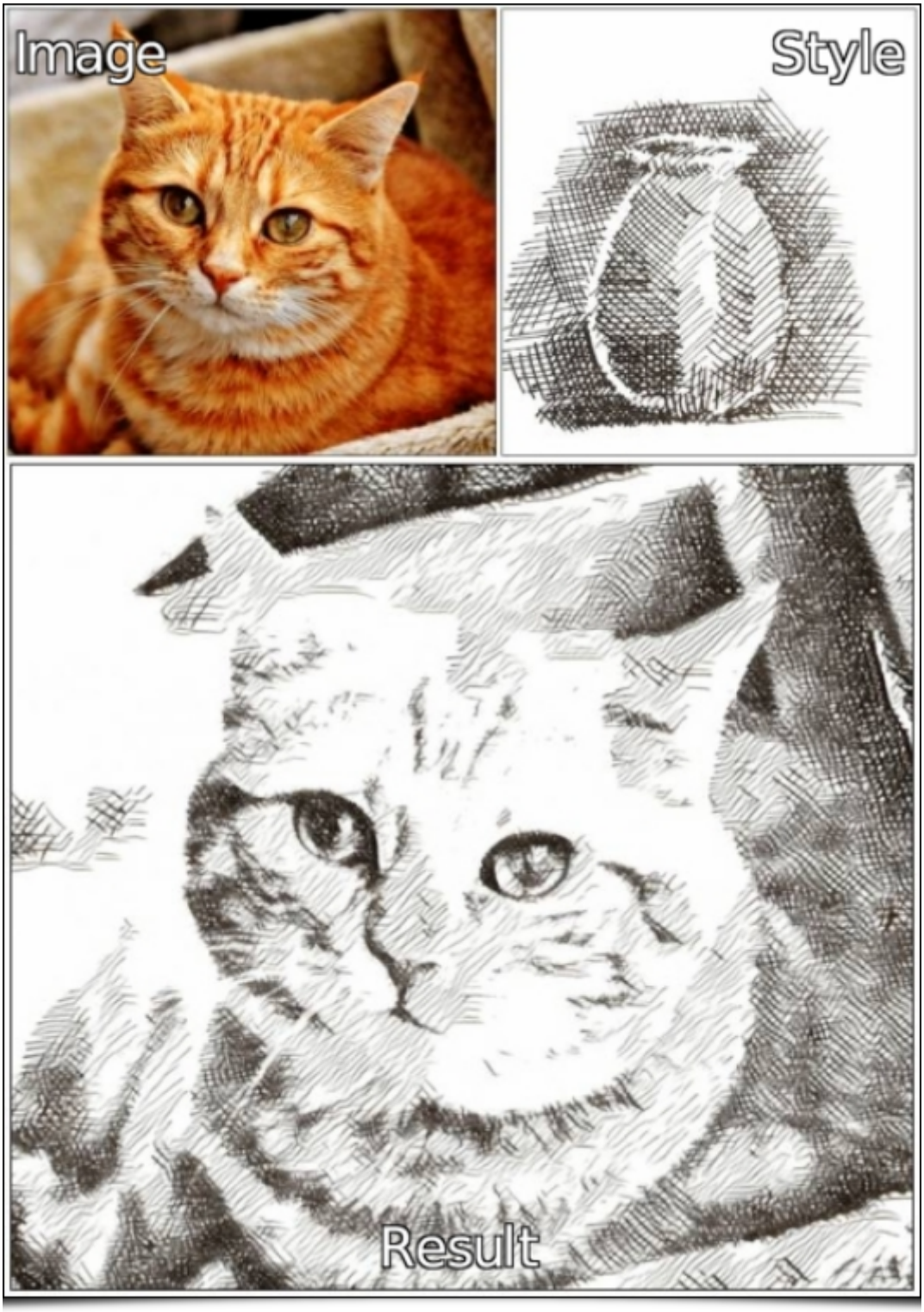


Fig.2.6: Stylization of a cat from a hatched drawing.

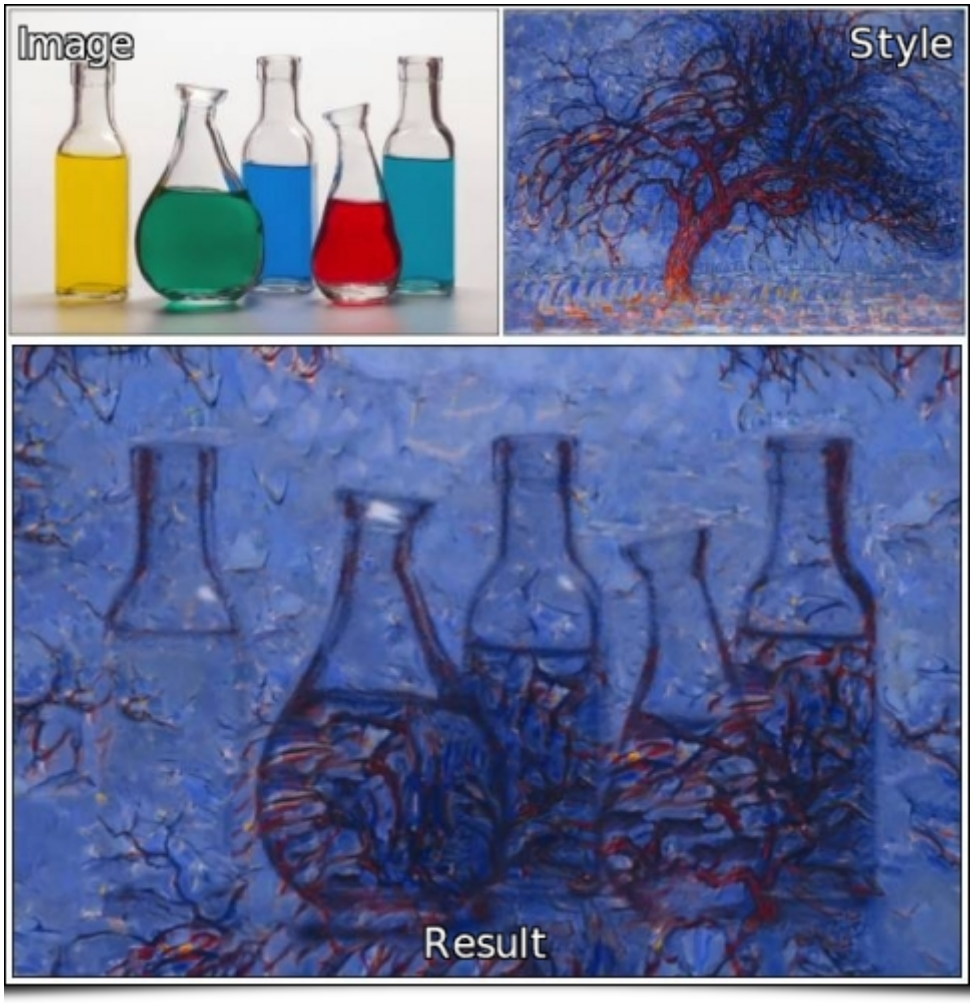


Fig.2.7: Stylization of bottles from the painting "Evening: Red Tree" by Piet Mondrian.

Fig.2.8: Stylization of bottles from the painting "Le réservoir - Horta de Ebro" by Pablo Picasso.

Other examples of image stylization can be found on the image gallery, dedicated to this filter. To our knowledge, G'MIC is the only "mainstream" image processing software currently offering a **generic** style transfer filter, where **any** style image can be chosen.

A last funny experiment: get a picture of an Alien's head, like *Roswell*, and then select a crop of the Mandelbrot fractal set as your style image. Use the transfer filter to generate a "fractal" rendering of your alien head. Then, make the whole world believe that the Mandelbrot set contains the mathematical proof of the existence of aliens... ☺

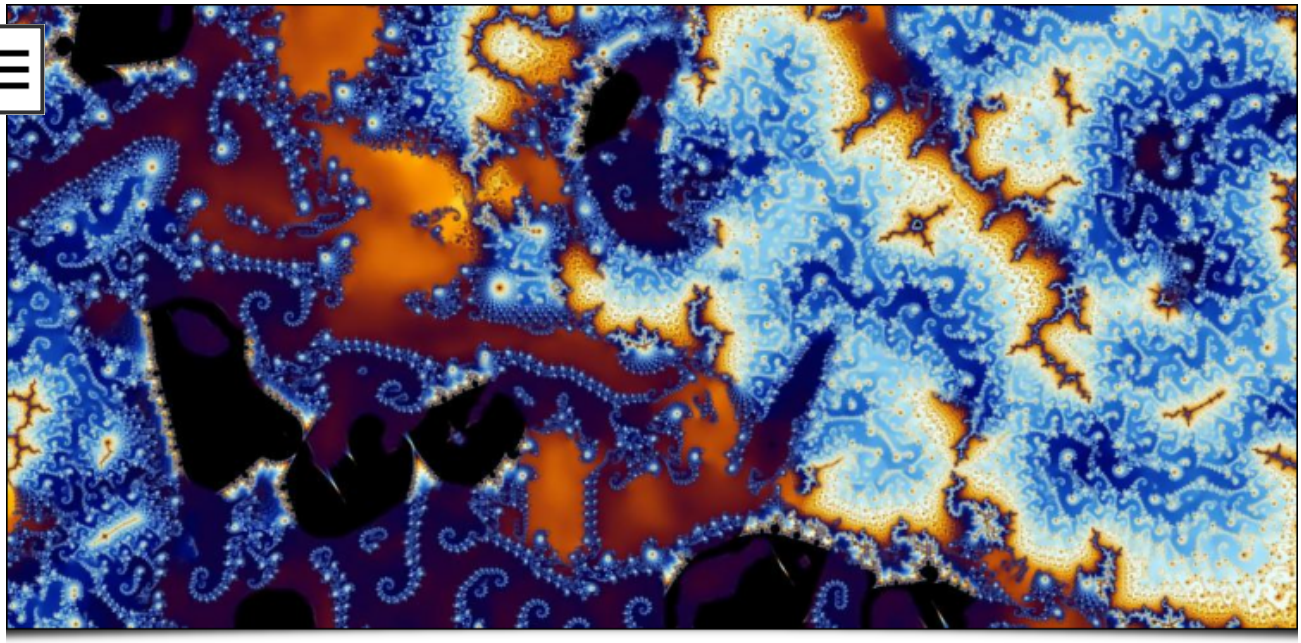


Fig.2.9: ***Breaking News!*** *An Alien head was found in the Mandelbrot fractal set ! (if you don't see it at first sight, tilt your head to the left...)*

In short, this filter has a clear creative potential for all kind of artists!

3. Interactive deformation and morphing

This year, *G'MIC* got an implementation of the *RBFs* interpolation method (*Radial Basis Functions*), which is able to estimate a dense interpolated function in any dimension, from a known set of scattered samples (not necessarily located on a regular grid). Thus, it gave us the idea to add distortion filters where the user interaction is focused in adding and moving keypoints over the image. In a second stage, *G'MIC* interpolates the data represented by these keypoints in order to perform the distortion on the entire image.

Let us start with the **"Deformations / Warp [interactive]"** filter which, as its name suggests, allows the user to distort an image locally by creating/moving keypoints.

Fig.3.1: The new ***"Deformations / Warp [interactive]"*** *filter allows images to be distorted interactively, for example to quickly create caricatures from portrait photographs.*

The animation below shows this interactive filter in use, and illustrates the fact that these keypoints can be considered as anchors to the image, when they are moved.

Fig.3.2: Illustration of the user interaction in the G'MIC deformation filter, based on the creation and motion of keypoints.

(For those who might be concerned about the portraits photos used in the figures above and below: all these portraits are totally

artificial, randomly generated by GANs via the website This Person Does Not Not Exist. No moral prejudices to dread!).

The great advantage of using RBFs-based interpolation is that we do not have to explicitly manage a *spatial structure* between the keypoints, for instance by defining a mesh (i.e. a "deformation grid"). This gives a greater degree of freedom in the obtained distortion (see *Fig.*3.3. below). And at the same time, we keep a rather fine control on the local amplitude of the applied distortion, since adding more "identity" keypoints around a region naturally limits the distortion amplitude inside this region.

*Fig.*3.3: *RBFs interpolation is able to create complex continuous distortions, with very few keypoints (here, by inverting the positions of the right/left eyes, and only 4 keypoints used).*

A short demonstration of this distortion filter is also visible in this Youtube video.

And why not extending this kind of distortion for two images, instead of a single one? This is precisely what the new filter **"Deformations / Morph [interactive]"** does. It is able to render a morphing sequence between two images (put on two separate layers), using the same interpolation technique that only asks for the user to set colored keypoints which match on both images.

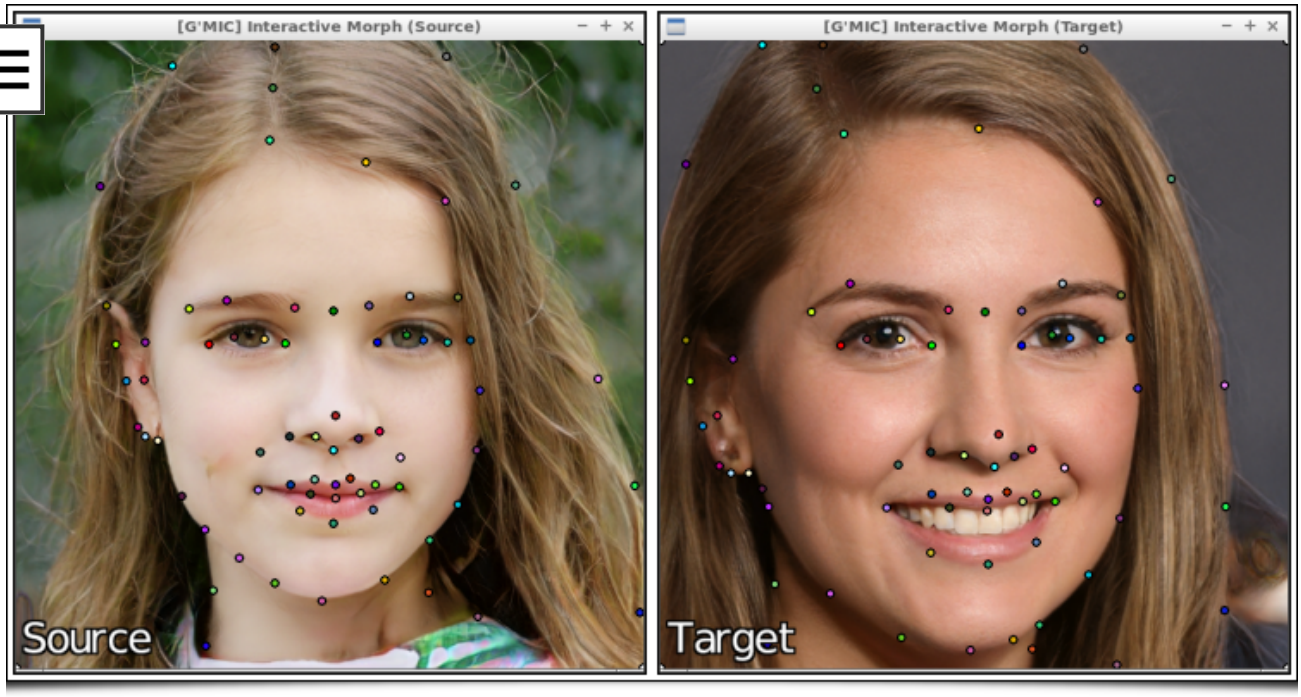


Fig.3.4: ***"Deformations / Morph [interactive]"*** *filter asks the user to position keypoints indicating correspondences between two images.*

In the example above, keypoints are placed on characteristic areas of both faces (tip of nose, lips, eyebrows, etc.). In practice, this takes no more than 5 minutes. Thanks to these keypoints, the algorithm is able to estimate a global deformation map from one image to another, and can generate temporally "mixed" frames where the elements of the face remain relatively well aligned during the whole morphing sequence.

Fig.3.5: One of the intermediate images generated by the morphing filter, between the two input faces.

By comparison, here is what we would obtain by simply averaging the two input images together, i.e. without correcting the displacement of the facial features between both images. Not a pretty sight indeed!

Fig.3.6: A simple averaging of the "Source" and "Target" images reveals the differences in the locations of the facial features.

Thus, the morphing filter is able to quickly generate a set of intermediate frames, ranging from the "Source" to the "Target" faces, a sequence that can then be saved as an animation.

Fig.3.7: Animation resulting from the generation of all intermediate frames by the G'MIC morphing filter.

Many other use cases of this morphing filter can be considered. The following example illustrates its application to render an animation from two photographs of the same object (a garden gnome), but shot with different DOFs (Depth of Field).

Fig.3.8: Two photographs with different depths of field, and the location of the correspondence keypoints put by the user.

Fig.3.9: Animation resulting from the generation of all intermediate frames by the G'MIC morphing filter.

Command line users will be pleased to know that these two filters can be tested very quickly from a *shell*, as follows:

```
$ gmic image.jpg x_warp
$ gmic source.jpg target.jpg x_morph
~
```

4. Ever more colorimetric transformations

For several years, G'MIC has contained colorimetric transformation filters able to simulate the film development process, or to give particular colorimetric moods to images (sunlight, rain, fog, morning, afternoon, evening, night, etc.). In a previous report, we already mentioned these filters, which are essentially based on the use of 3D CLUTs (*Color Lookup Tables*) for modeling the color transformation.

A 3D CLUT is technically a three-dimensional array that provides for each possible RGB color, a replacement color to apply to the image.

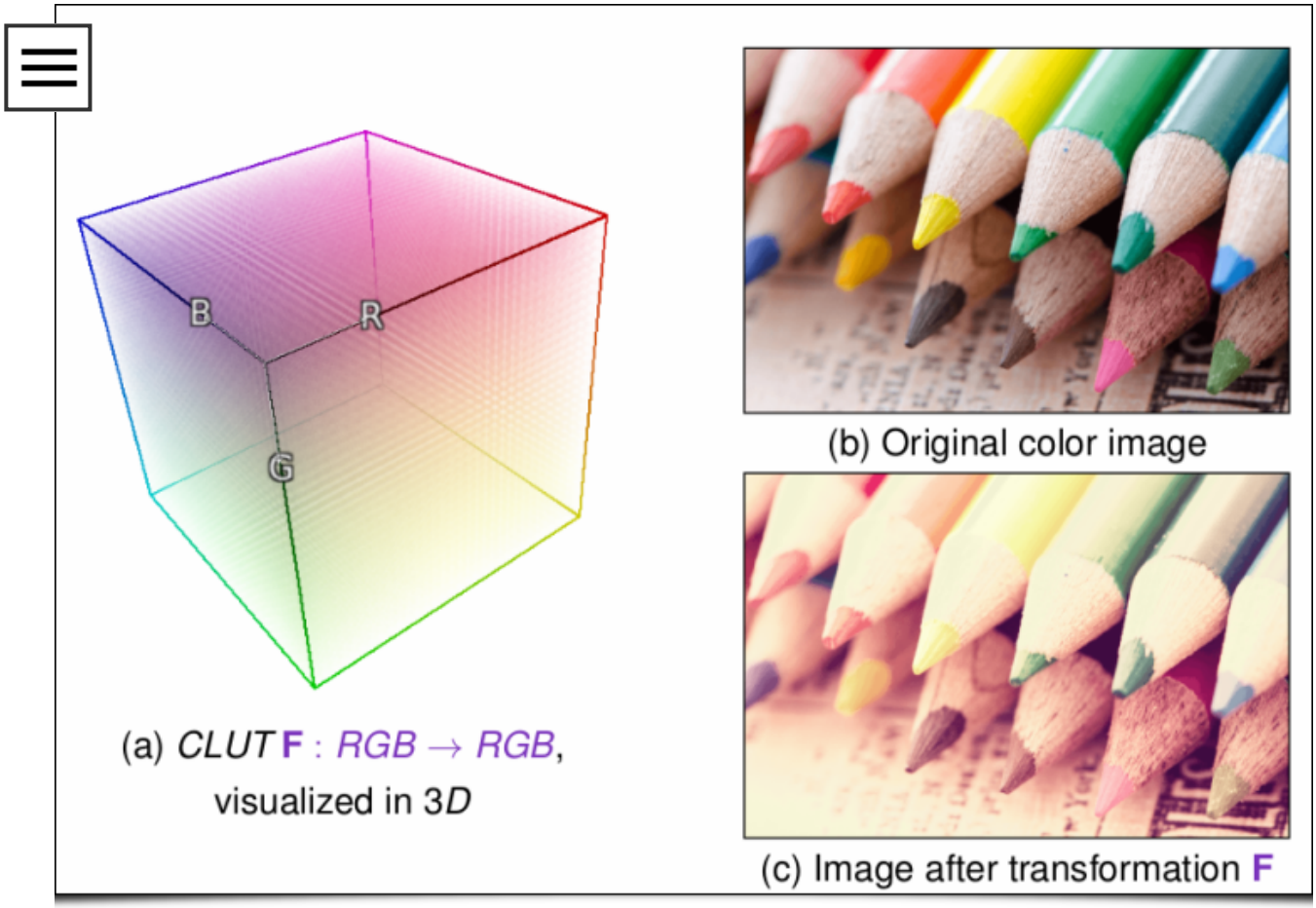


Fig.4.1: Modeling a colorimetric transformation by a "3D Color LUT".

The main interest of these 3D CLUTs is the great variety of transformations they can represent: They can indeed define RGB-to-RGB functions with almost any kind of variations. The only "constraint" of these methods is that all image pixels having the same color will be transformed into pixels that also have an identical color.

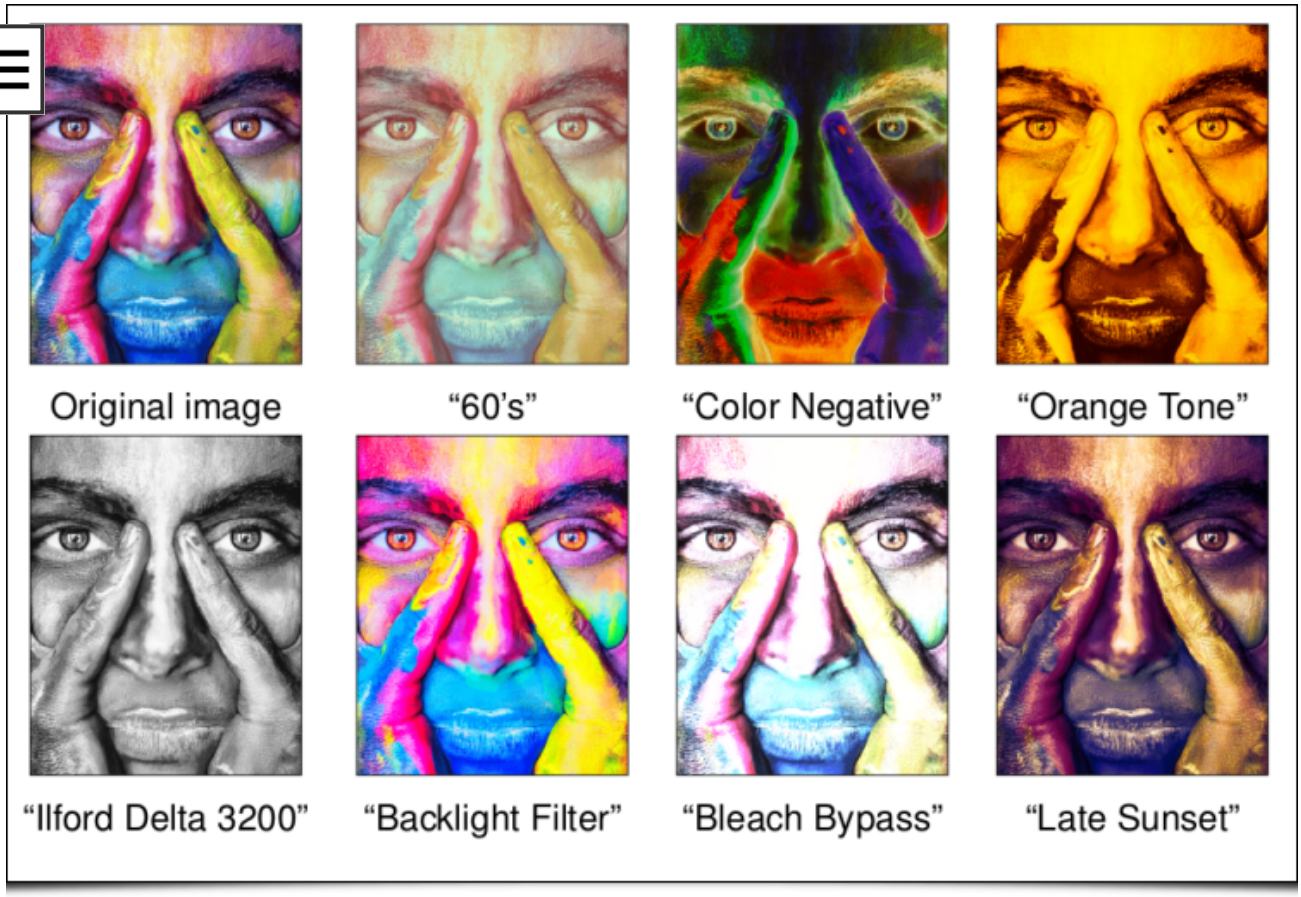


Fig.4.2: Illustration of the variety of colorimetric transformations that can be modeled by 3D CLUTs.

The disadvantage, however, is that these 3D *CLUTs* are relatively data intensive. When you want to embed several hundred different ones in the same piece of software (which is the case in *G'MIC*), you quickly find yourself with a large volume of data to install and manage. For instance, our friends at RawTherapee offer on their website an additional pack of **294** *CLUTs* functions to download. All these *CLUTs* are stored as `.png` files in a `.zip` archive with a total size of **402 MB**. Even if downloading and storing a few hundred _MB_ is no longer limiting nowadays, it is still quite large for things as simple as color changing filters.

This year, we have therefore carried out important research and development work at the GREYC lab on this topic. The result: a new lossy compression algorithm (*with visually imperceptible compression losses*) that can generate binary representations of *CLUTs* with an average compression rate **of more than 99%**, relative to the data already *loslessy compressed*. The general idea is to determine an optimal set of color keypoints from which the *CLUT* can be reconstructed (*decompression*), and this, with a minimal reconstruction error.

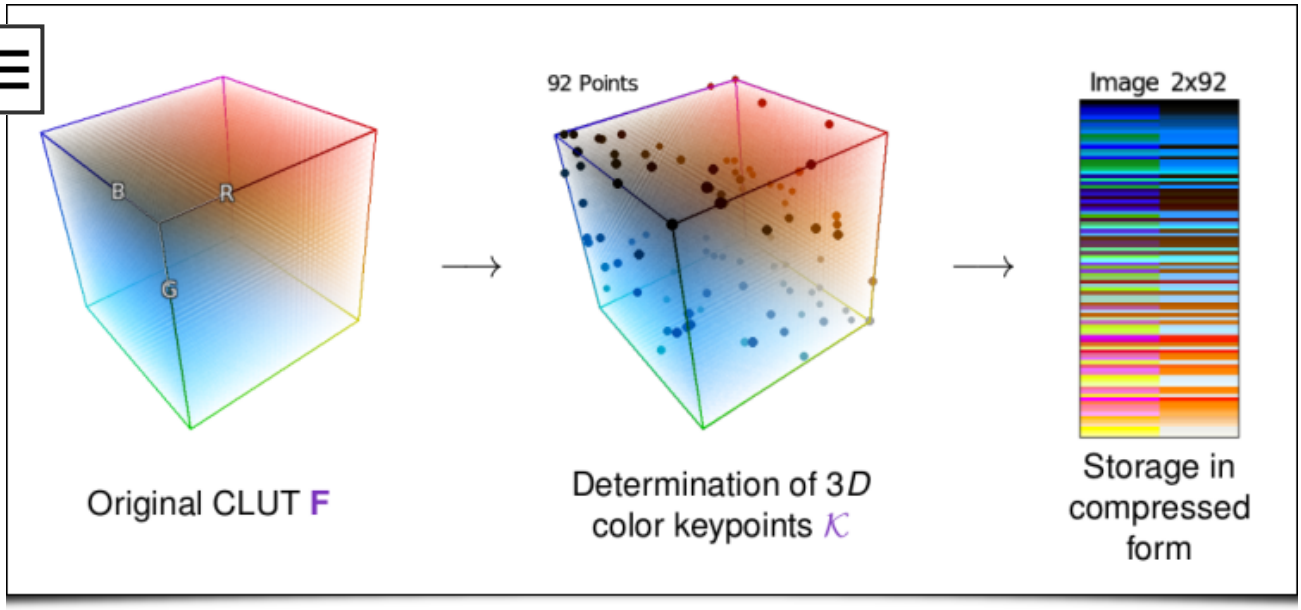


Fig.4.3: Principle of our CLUT compression technique, based on determining and storing a set of well-chosen keypoints.

As a result, this original compression method allowed us to offer no less than **763 CLUTs** in G'MIC, all stored in a binary file that weights **less than 3 MB** !

All these color variation filters have been grouped into two separate entries in the G'MIC-Qt plug-in, namely **"Colors / Simulate Film"** (for analog film simulations), and **"Colors / Color Presets"** (for other color transformations). Each of these filters provides sub-categories for a structured access to the hundreds of CLUTs available. To our knowledge, this makes G'MIC one of the image processing software with the most colorimetric transformations, while keeping a reasonable size.

Readers interested in the mathematical details of these CLUT compression/decompression algorithms may refer to the scientific paper we wrote about it, as well as the presentation slides that have been presented at the conferences GRETSI'2019 (French conference, in Lille) and CAIP'2019 (International conference, in Salerno).

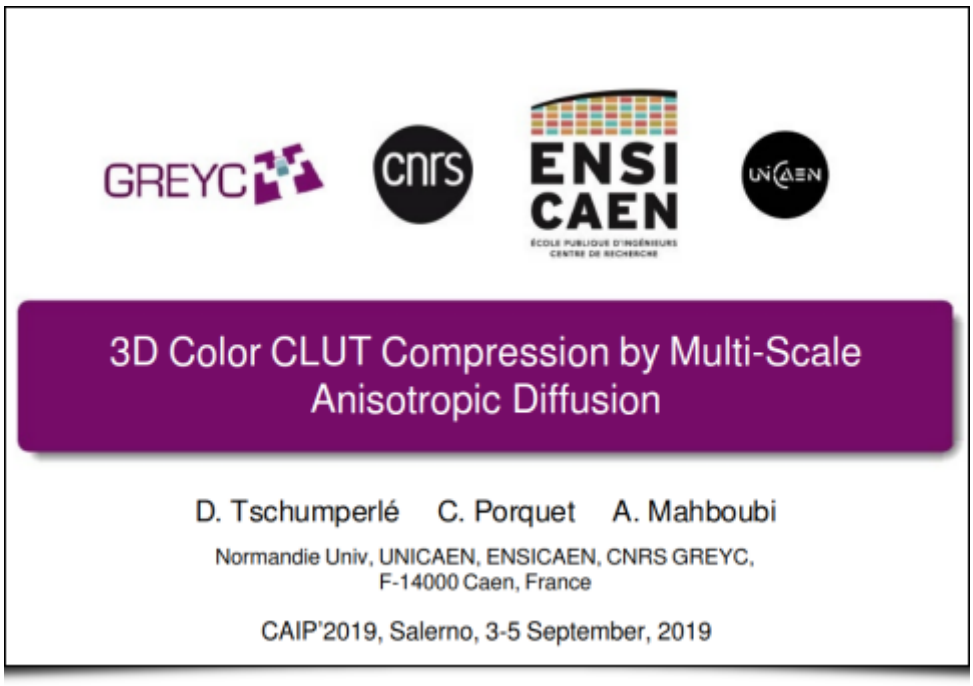


Fig.4.4: Presentation slides explaining the details of the CLUT compression/decompression algorithm.

To finish with this topic, note that we have made an open-source implementation of our decompression algorithm of *CLUTs* available online (in *C++*, with 716 *CLUTs* already included). Discussions have also been initiated for a potential integration as a Darktable module for managing *3D CLUTs*.

5. Create palettes by mixing colors

Let us now talk about the recent **"Colors / Colorful Blobs"** filter which is directly inspired by the original concept of *Playful Palette* created by the Adobe Research team in 2017. This filter is intended for illustrators (designers and digital painters). The goal: Create color palettes which contain only a few main colors (the ones you want to use in an illustration), but also a few sets of intermediate shades between these colors, in the form of color gradients. An artist is theoretically able to better preserve the color coherence of its artwork, by picking colors only from this palette.

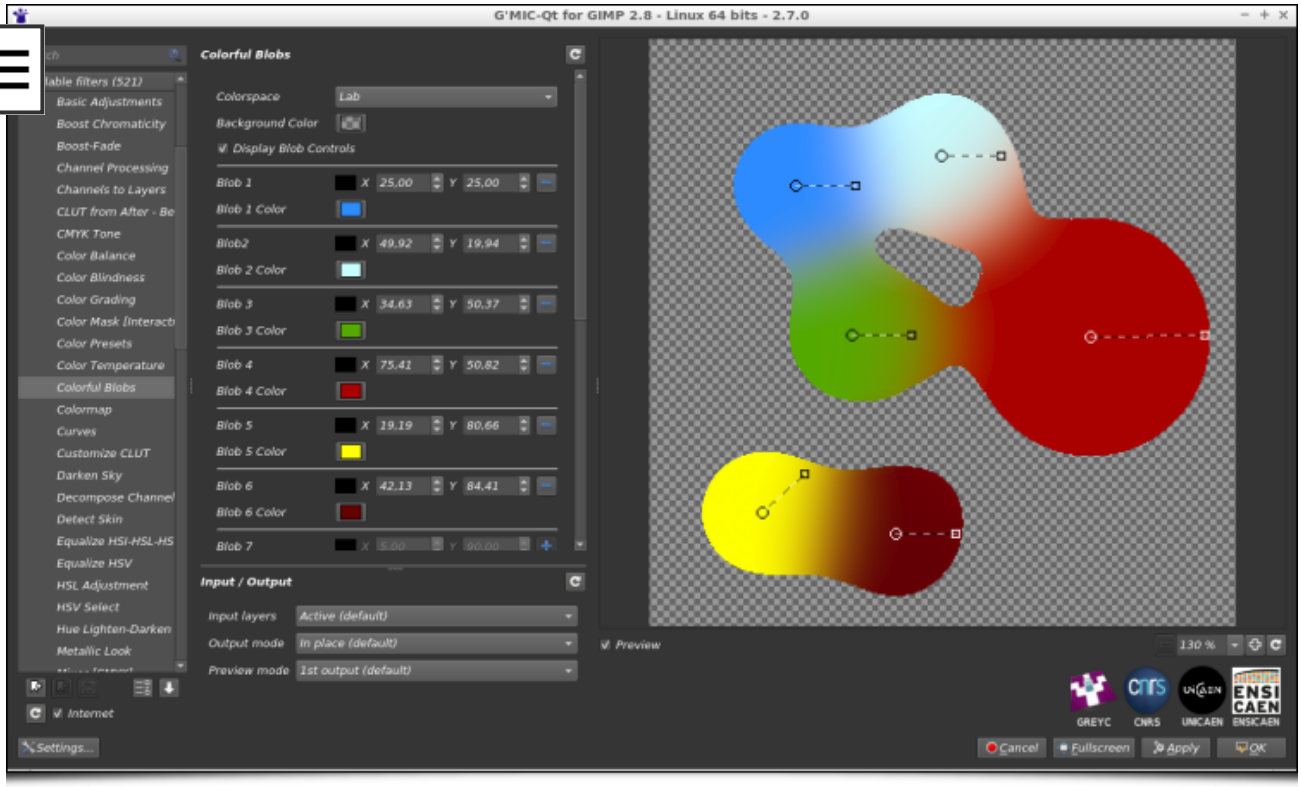


Fig.5.1: ***"Colors / Colorful Blobs"*** *filter allows you to create custom color palettes, by spatially mixing several colors together.*

As shown on the figure above, the filter allows the artist to create and move colored "blobs" that, when merged together, create the desired color gradients. The result of the filter is thus an image that the artist can use afterward as a custom 2D color palette.

From a technical point of view, this filter is based on 2D *metaballs* to model the color blobs. Up to twelve separate blobs can be added and different color spaces can be chosen for the calculation of the color gradient (*sRGB*, *Linear RGB* or *Lab*). The filter also benefits from the recent development of the *G'MIC-Qt* plug-in that enhances the user interactivity inside the preview widget (a feature we mentioned in a previous report), as seen in the animation below (see also this longer video).

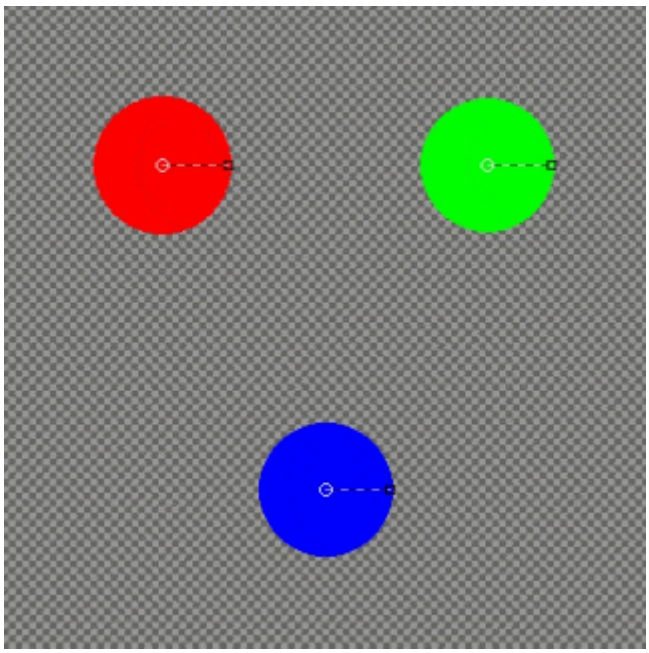


Fig.5.2: Illustration of the user interaction with the G'MIC palette creation filter, based on the creation and movement of colored "blobs".

This filter may not be useful for most G'MIC users. But you have to admit, it's pretty fun, isn't it?

6. Some more filters

Let us now describe a selection of a few other filters and effects added during the year, perhaps less original than the previous ones (but not completely useless anyway!).

- First of all, the **"Rendering / Symmetric 2D Shape"** filter is a great help when you want to draw geometric shapes having angular symmetries.

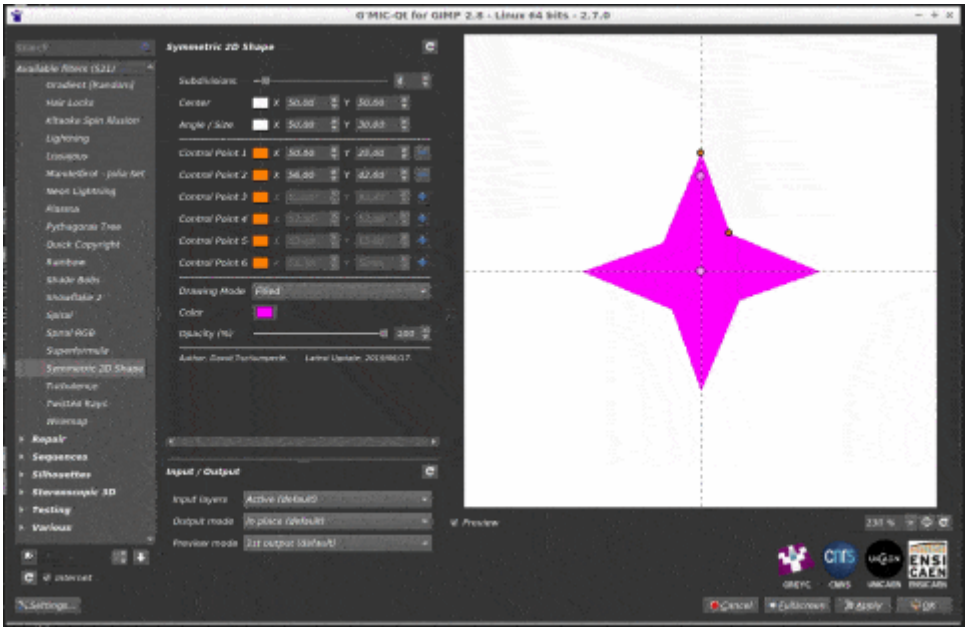


*Fig.6.1: **"Rendering / Symmetric 2D Shape"** filter in action, in the G'MIC-Qt plug-in.*

The plane can be subdivided into up to 32 angular pieces, each of which can contain a maximum of six keypoints to define a shape profile, allowing potentially complex and varied shapes to be rendered (such as the super-shuriken below!).

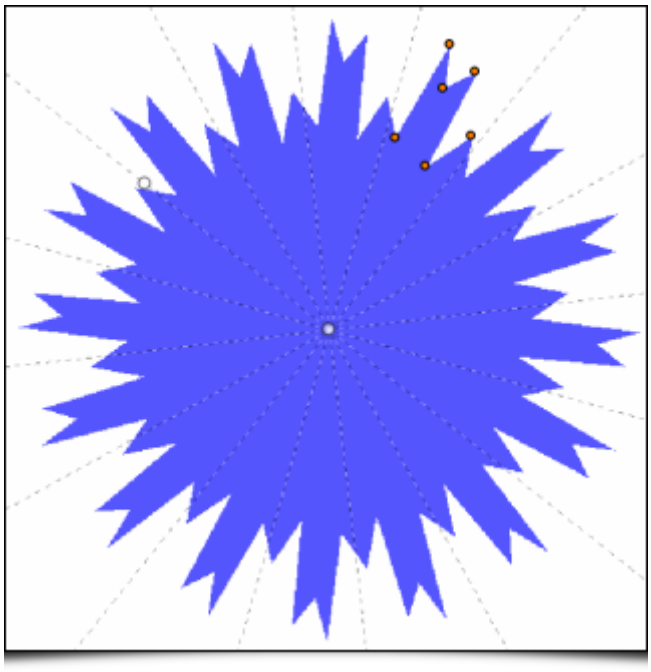


*Fig.6.2: Example of a complex symmetrical shape obtained with the **"Rendering / Symmetric 2D Shape"** filter.*

- The **"Degradations / Self Glitching"** filter combines an image with a shifted version of itself, to create a *Glitch-art* type image. Several bitwise operations (*Add*, *Mul*, *And*, *_Or_*, *Xor*,...) can be chosen and you can adjust the shift direction and amplitude, as well as various other controls.

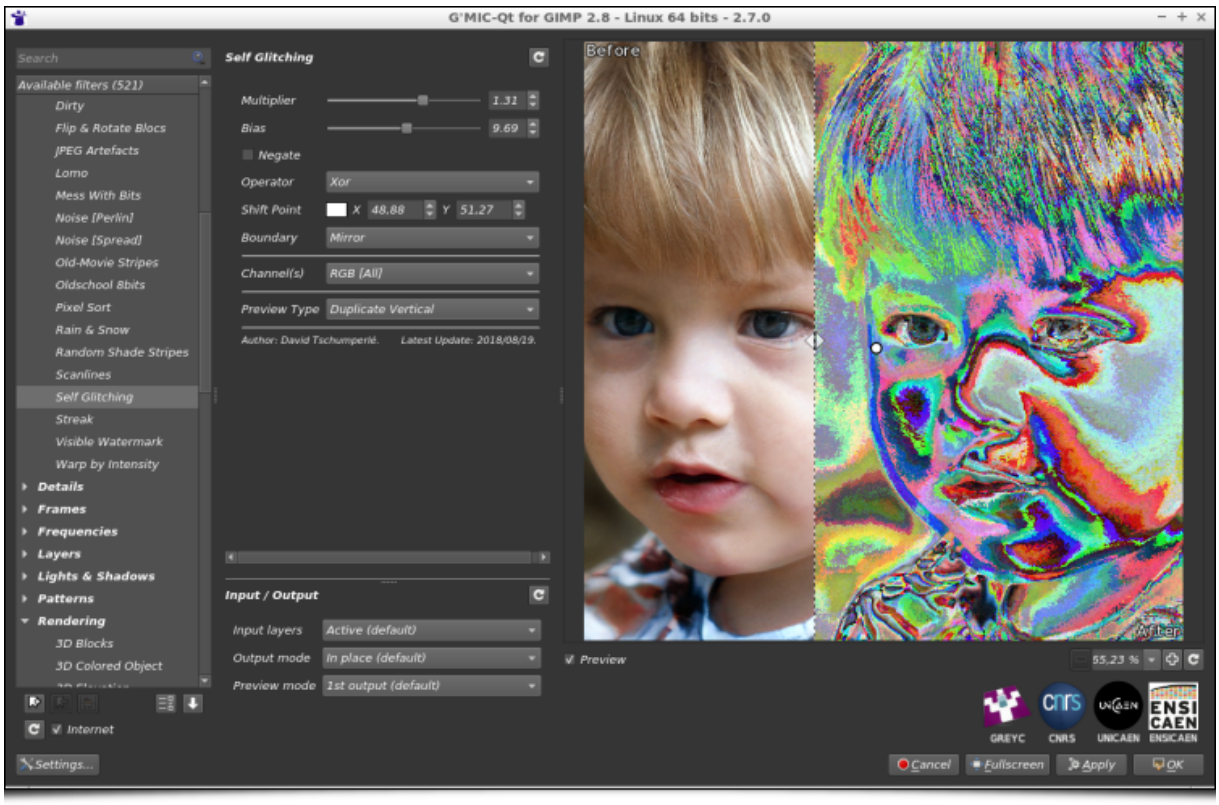


Fig.6.3: ***"Degradations / Self Glitching"*** *filter helps to ruin your photos easily!*

Again, this is not a filter that will necessarily be used every day! But it may be helpful for some people. It was actually added in response to a user request.

- In the same style, the **"Degradations / Mess With Bits"** filter applies some arithmetic operations to the pixel values, seen as binary numbers (for instance, bit shift and bit inversion). Always with the idea of rendering *Glitch art*, of course!

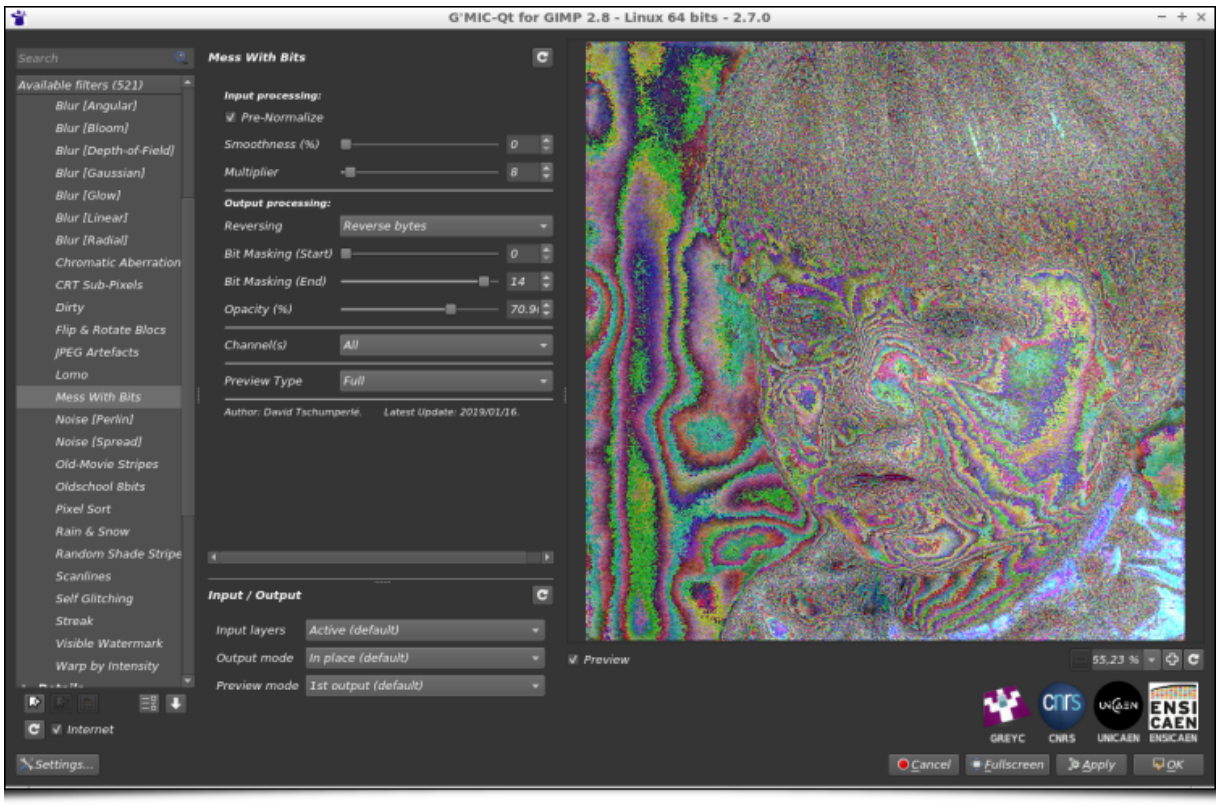


Fig.6.4: ***"Degradations / Mess With Bits"*** *filter, or how to transform an adorable toddler into a pustulating alien...*

- The **"Degradations / Noise [Perlin]"** filter implements the generation of the Perlin noise, a very classical noise model in image synthesis, used for the generation of elevation maps for virtual terrains. Here we propose a multi-scale version of the original algorithm, with up to four simultaneous variation scales.

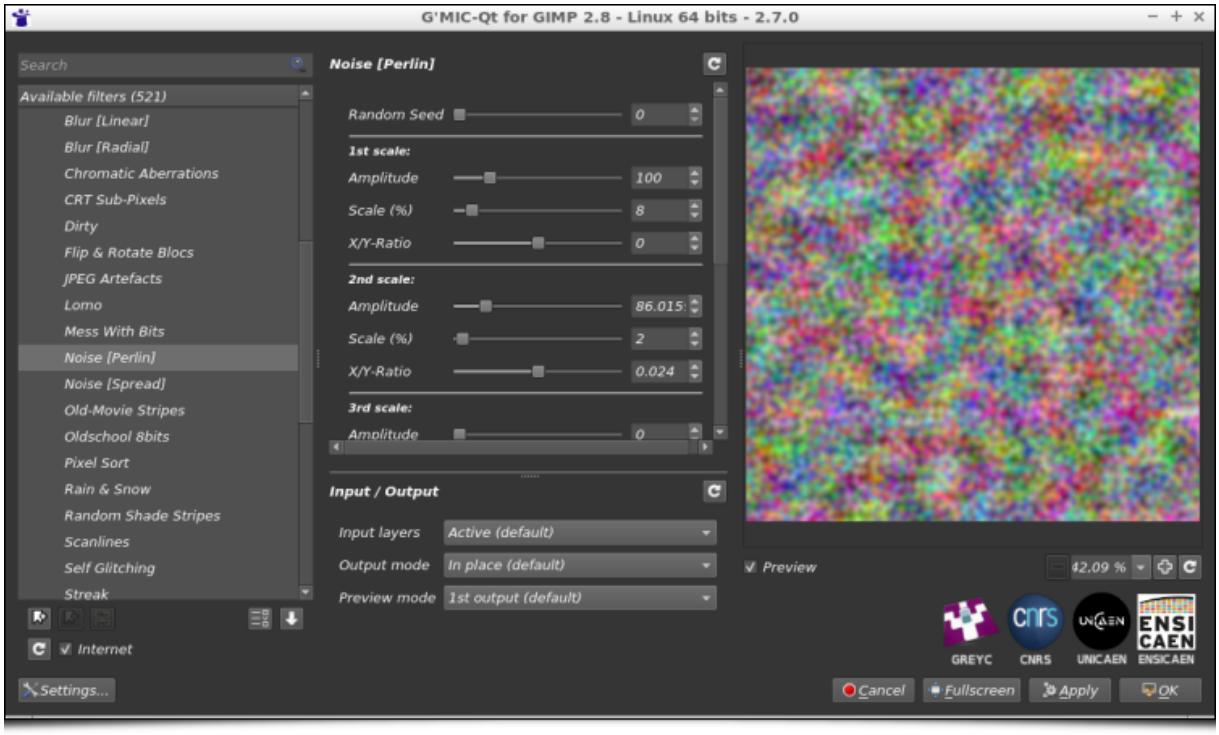


Fig.6.5: ***"Degradations / Noise [Perlin]"*** *filter proposes a multi-scale implementation of the Perlin noise (illustrated here with two variation scales).*

- The **“Frames / Frame [Mirror]”** filter is also a “tailor-made” effect, to meet the needs of a *G’MIC-Qt* plug-in user. This photographer wanted to resize his photos to obtain a precise *width/height* ratio, but without having to crop his images. The solution was instead to add image information at the edges of the picture, by symmetry, in order to obtain the desired ratio. So that’s what this filter does.

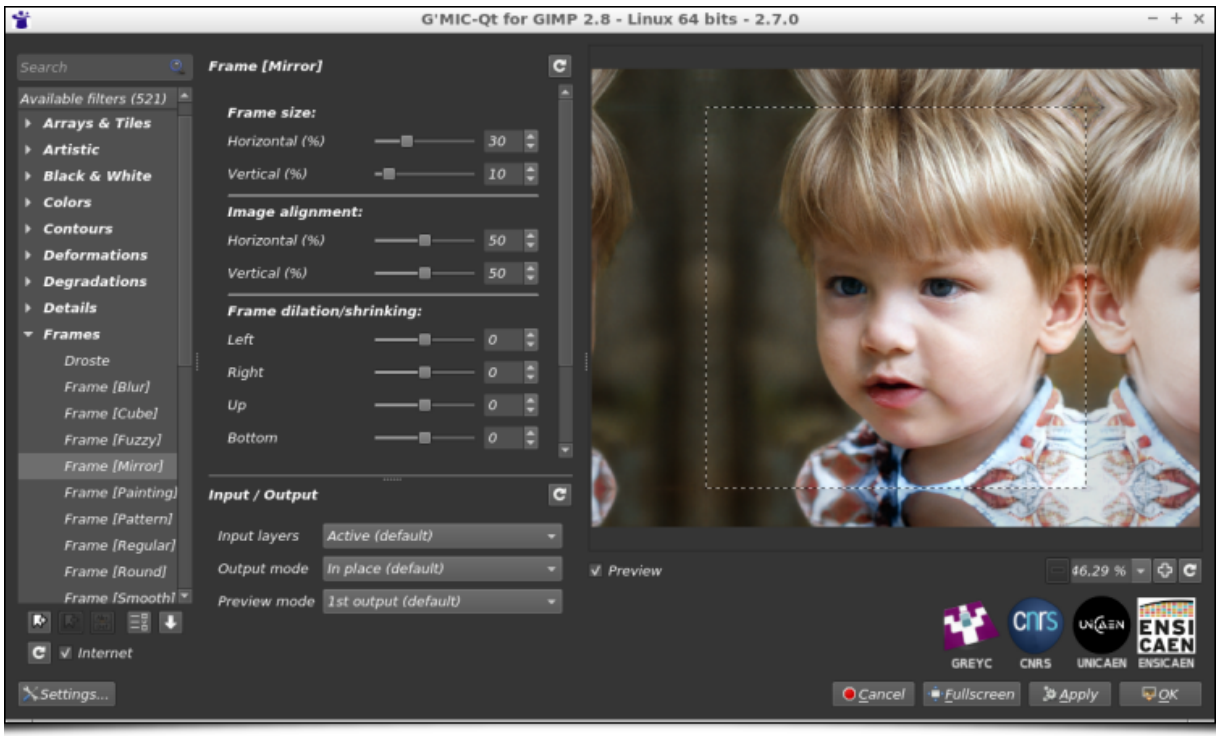


*Fig.6.6: The **“Frames / Frame [Mirror]”** filter extends the image borders by symmetry.*

- Finally, let us mention the upcoming advanced image noise reduction filter, by Iain Fergusson, whose development is still in progress. Iain has been contributing to *G’MIC* for several years now by implementing and experimenting original denoising filters, and his latest project seems really interesting, with promising results. This video shows this filter in action, a good place to learn a little more about how it works.

Now that we’ve looked at these new filters, it seems important for us to remind that, as in many IT projects, this visible part of the iceberg hides a set of lower-level developments done to improve the interactive possibilities of the *G’MIC-Qt* plug-in, as well as the performance of the internal scripting language interpreter (the *G’MIC* language), which is how all these filters and effects are actually implemented. These improvements and incremental slight optimizations of the code base benefit to all filters (even those already available for several years) and it actually represents most of the development time we spend on

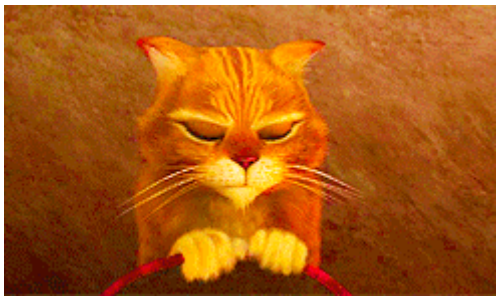
G'MIC. So, dear users, do not be surprised if no new filters appear for a while. It is probably just because we are doing serious work on the G'MIC framework core!

7. Other notable points in the project life

Here are listed some other important news that have punctuated the life of the project since August 2018.

7.1. We now accept donations!

This is essential news for us: since March 2019, the G'MIC project has been granted permission to **collect donations** (via *Paypal*), to help in its maintenance and development!



This is a good thing, because until now, there was no simple way for a public research laboratory as the GREYC, to accept donations for supporting the development of a free software application such as G'MIC, an application used daily by several thousand people around the world. And we have currently no other ways to finance this piece of software in the long term.

Thus, we have partnered with LILA (*Libre comme l'Art*), a French non-profit organization promoting Arts, Artists and Free Software, who accepted to collect donations for us.

Fig.7.1: Logo of the LILA association, which collects donations for the G'MIC project.

In practice, this is something that has been a little long to set up, but now that the donation system is operational, we hope to benefit from it in the future to make the project development even faster (the possible use of the raised funds is detailed on the donations page, this being of course very dependent on the amount of money collected).

For the sake of transparency, we will post the monthly amount of collected donations on the project website. At this point, we don't really know what to expect in practice. We will see how these donations evolve. Of course, we would like to thank all those who have already participated (or plan to do so) in supporting our open-source framework for image processing. Our ultimate dream would be, one day, to say that the illustration below is only a distant memory!

Fig.7.2: The harsh reality of the development of the G'MIC project ☺ (illustration from the CommitStrip website).

7.2. Integrating "Smart Coloring" into GIMP

Let us also mention the work of Jehan, known to PIXLS.US readers as a regular GIMP developer. Jehan has been hired by the GREYC laboratory in September 2018, to work on G'MIC (for a 12-month fixed-term contract), thanks to a grant funded by the INS2I Institute of the CNRS (for which we are grateful).

One of its first missions was to re-implement the *G'MIC* "Smart Coloring" algorithm (that we had already talked about previously) as a new interactive mode integrated into the existing GIMP "*Bucket Fill*" tool.

Fig.7.3: G'MIC's "Smart Coloring" algorithm, now available in GIMP, helps illustrators color their drawings more quickly.

Jehan described all his work in a blog post, which is strongly recommended for reading. Of course, we don't want to copy his post here, but we want to mention this activity, and to consider it as another original contribution of the *G'MIC* project to free software for graphic creation: at the GREYC laboratory, we are really happy and proud to have imagined and developed an image colorization algorithm, which artists can use through a well integrated tool into such a popular piece of software as GIMP!

This intelligent colorization algorithm has been the subject of scientific publications, presentations at the conferences *GRETSI'2017*, *EuroGraphics VMV'2018*, as well as at the *Libre Graphics Meeting'2019*. And it is with a great pleasure we see this algorithm is used in real life, for various realizations (as in this great video of *GDQuest*, for colorizing sprites for video games, for instance).

Scientific research carried out in a public laboratory, which becomes available for the general public, that is what we want to see!

7.3. Other news related to the *G'MIC* project

- Recently, a major improvement in the performances of *G'MIC* under *Windows* has been achieved, by recoding the random number generator (now reentrant)) and removing some slow

mutex which were responsible of performance drops for all filters requiring sequences of random numbers (and there were many!). As a result, some filters are accelerated by a factor of four to six under Windows!

- Since December 2018, our *G'MIC-Qt* plug-in is available for *Paint.net*, a free graphic editing software application under *Windows* (not open-source though). This has been possible thanks to the work of Nicholas Hayes who wrote the glue code allowing the interaction between our *G'MIC-Qt* plug-in and the host software. Users of Paint.net are now able to benefit from the 500+ filters offered by *G'MIC*. This plug-in, available here, has already been voted "*Best Plug-in of the Year 2018*" by the members of the *Paint.net* forum ☺ !

- Since October 2018, the *G'MIC-Qt* plug-in for GIMP has been compiled and proposed for *MacOS* by a new maintainer, Andrea Ferrero, who is also the main developer of the free software application Photoflow, a non-destructive image editor (more information here). Many thanks Andrea, for this wonderful contribution!

  - Since the announced shutdown of the *Google+* social network, we have opened two new accounts, on Framasphere and Reddit, to share news about the project's life (but the Twitter feed is still our most active account).

- Let us also thank *Santa Claus*, who kindly brang us a materialized version of our mascot "Gmicky" last year. That looks almost perfect!

Fig.7.4: The mascot "Gmicky", brought by Santa Claus, in December 2018.

- The G'MIC project was presented at the FENO, the "*Fête de l'Excellence Normande*", from 12 to 14 April 2019, at the Caen Exhibition Centre. We were hosted on the stand of the CNRS *Normandie*, and we carried out demonstrations of style transfer (*teaser*) and automatic illumination of clip arts (*teaser*), for the general public.

Fig.7.5: We were present at the CNRS stand, for G'MIC demonstrations, at the "Fête de l'Excellence Normande 2019" (FENO).

- And to dig even deeper, here are some other external links we found interesting, and which mention G'MIC in one way or another:
  - A video presentation of the plug-in G'MIC-Qt, by *Chris' Tutorial*;
  - The Youtube channel *MyGimpTutorialChannel* offers a lot of videos showing how to use G'MIC-Qt in GIMP to achieve various effects (mostly in

German);

- *The Clinic*, a Chilean weekly newspaper, apparently used *G'MIC* to achieve an effect on one of its covers (via the smoothing filter **"Artistic / Dream Smoothing"**);
- Another video tutorial, showing how to use the *G'MIC* **"Artistic / Rodilius"** filter to create stylized animal photos.

8. The future

As you see, *G'MIC* is still an active open-source project, and with its 11 years of existence, it can be considered as mature enough to be used "in production" (whether artistic or scientific).

We have never defined and followed a precise roadmap for the project development: the functionalities come according to the needs of the developers and users (and the limited time we can devote to it!). At the moment, there is a lot of interest in image processing methods based on neural networks, and deep learning techniques. It is therefore possible that one day, some of these methods will be integrated into the software (for instance, we already have a prototyped code running in *G'MIC* that actually learns from image data with convolutional neural networks, but we are still at the prototyping stage...).

After 11 years of development (make it 20 years, if we include the development of the *CImg* library on which *G'MIC* is based), we have reached a point where the core of the project is, technically speaking, sufficiently well designed and stable, so as not to have to rewrite it completely in the next years. In addition, the number of features available in *G'MIC* already covers a large part of the traditional image processing needs.

The evolution of this project may therefore take several paths, depending on the human and material resources that we will be able to devote to it in the future (for the development, but also in project management, communication, etc.). Achieving an increase in these resources will undoubtedly be one of the major challenges of the coming years, if we want *G'MIC* to continue its progress (and we already have plenty of ideas for it!). Otherwise, this image processing framework might end up being just maintained in its current (and functional) state. It is of course with a hope for progression that we have recently set up

the donation page. We also hope that other opportunities will soon arise to enable us to make this project more visible (you are invited to share this post if you like it!)

That's it for now, this long post is now over, thank you for holding on until the end, you can resume normal activity! I'll be happy to answer any questions in the comments.

Post-scriptum: Note that the 3D animation displayed as the *teaser* image for this post has been actually generated by G'MIC, via the command `$ gmic x_starfield3d`. An opportunity to remind that G'MIC also has its own _3D_ rendering engine capable of displaying simple objects, which is very practical for scientific visualization! We may have the occasion to talk about it again in a future post...

A special thank you for reviewing and helping to translate this article to:
Patrick David, Sébastien Fourey, Christine Porquet, Ryan Webster.

WRITTEN BY:

David Tschumperlé

I am David Tschumperlé, a permanent researcher working in the field of image processing in a daily basis, since 1999. I work for the CNRS institute, more particularly in the Image Group at the GREYC laboratory in Caen/France.

←PREVIOUSLY
Quick digiKam Tip: Back up digikamrc file