# Retrogressive "Classic Game" Image Processing

Daniel R. Page, Neil D. B. Bruce*

Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada
drpage@pagewizardgames.com, bruce@cs.umanitoba.ca
October 2014

**Abstract**

Image processing has been applied for aesthetic or artistic purposes to produce a range of visual effects including abstracted images, painterly renderings, and comic-book, or cartoon effects. In this paper we examine the problem of transforming standard RGB images to having an appearance reminiscent of older console games. This is achieved by way of quantization of the colour-space, accompanied by a number of complementary processing stages that optimize the spread of initial RGB values for the target palette. Specific palettes considered include the Nintendo Entertainment System, Nintendo Gameboy, Sega Master System, Super Nintendo Entertainment System, and TurboGrafx–16. Methods and experimentation presented reveal that distinct strategies are required dependent on properties of source images, and target palettes. Observations driven by experimentation are provided as guidelines for quickly adapting source imagery to the appearance of a desired classic console game, and accompanying software is provided with an overview of its use case.


Completed: October 2013 Published: October 2014

keywords: color, quantization, aesthetic, stylistic, video games, median cuts

## I. Introduction

There have been a numerous efforts in the past 10-15 years aimed altering images for aesthetic reasons, including creating cartoon-like images [10, 11], painterly renderings [4, 5], or making an image appear as though it was captured by an alternative camera or imaging system [6]. This is often done for nostalgic or aesthetic reasons, and image effects and filters are readily available in application software for smart cameras and phones. In the current work, we consider altering the characteristics of images towards having an appearance consistent with imagery from a number of classic video game consoles. This has value to both aesthetic and nostalgic appeal, and there remains a sizeable niche of developers, and hobbyists interested in emulating a *classic* look in video games. The historical importance of graphics from classic games is also apparent in efforts aimed at methods to best preserve knowledge of standards in graphics subject to evolution over time [2].

In this paper, we present methods for achieving a mapping from RGB images, to a chromatic representation consistent with imagery corresponding to classic console games. The paper is structured as follows: First the details of the various classic game palettes are described. Following this, we describe the RCGIP method, along with details of optional operators that aid in producing

*The authors gratefully acknowledge the financial support of the University of Manitoba and NSERC.

| Console | Colorspace | # Colors | Notes |
|---|---|---|---|
| NES | YPbPr | 64 | the 55 unique colors are used |
| GB | Mono | 4 | Also emulated *greenish* display. |
| SMS | 6 bit RGB | 64 | 2 bits per channel. |
| SNES | 15 bit RGB | 32768 | 5 bits per channel. |
| TG16 | 9 bit RGB | 512 | 3 bits per channel. |

**Table 1:** *Different console palettes and their properties. Note that for the NES, many colors consist of a repeated black, and only 55 distinct colors were used. For the GB monochrome, intensities were matched to the green of the Nintendo Gameboy display for more faithful reproduction of the visual appearance of Gameboy graphics.*

high quality results given the properties of source images, and target palette. Following this, software that accompanies this work (the RCGIP software) is discussed, and typical use cases given. It is important to note that given specific source imagery, and a target palette, there is generally not a *one size fits all* approach to achieving the most desirable transformation. For this reason, the software is included as one contribution of this paper, and discussion in the paper is targeted towards strategies that allow one to apply the methods described to quickly achieve desirable results.

## II. Classic Game Palettes

The RCGIP method has been applied with consideration to five different classic game console colour palettes. The aim of this work is to effectively re-map an image to the color palette of a console system in a fashion that optimizes the fidelity of the end result. An important detail in re-mapping of colors, is the specifics of palletes used in console games. Details of all of the classic game color palettes considered, and additional notes appear in Table 1. Abbreviations used in Table 1 are as follows: *Nintendo Entertainment System* (**NES**), *Nintendo Gameboy* (**GB**), *Sega Master System* (**SMS**), *Super Nintendo Entertainment System* (**SNES**), *Turbographx–16* (**TG16**).

## III. The RCGIP Algorithm

The basic structure of the RCGIP processing pipeline is presented below, followed by further detail on the various processing stages. Re-mapping the color-space from a native RGB image to an indexed console color-space is achieved via a heuristic algorithm based on Median Cuts [7]. The median cut algorithm is one of the more successful approaches to color quantization and has also been applied successfully to simulate lighting conditions in computer graphics [1] as well as general multi-dimensional data clustering [9]. In the Median Cuts Algorithm [3], data is sorted into a series of sets by dividing the data based on the median value. Initial RGB values are divided into two sets based on a median cut of the RGB vectors corresponding to each pixel location. Subsequently the largest subset is again divided by Median cuts. This process is repeated until the number of subsets matches the dimensionality of the quantized console color space. A centroid for each subset establishes a mapping to the closest color in the quantized palette.

The RCGIP algorithm has 5 stages. These steps are applied to the input image $I$. It is worth noting that many of the operations listed have an (O) tag associated with them indicating that these are optional operations in the pipeline. This reflects the fact that in our experimentation, there was not a general principle that could be applied broadly to all of the categorical variants for source images, and all of the console palettes. For this reason, in the discussion of the details of the various operations, emphasis is placed on instances where these operations demonstrated success in producing a desirable outcome. The 5 stages involved in the RCGIP pipeline are as follows:

1. Downsample $I$ (O),

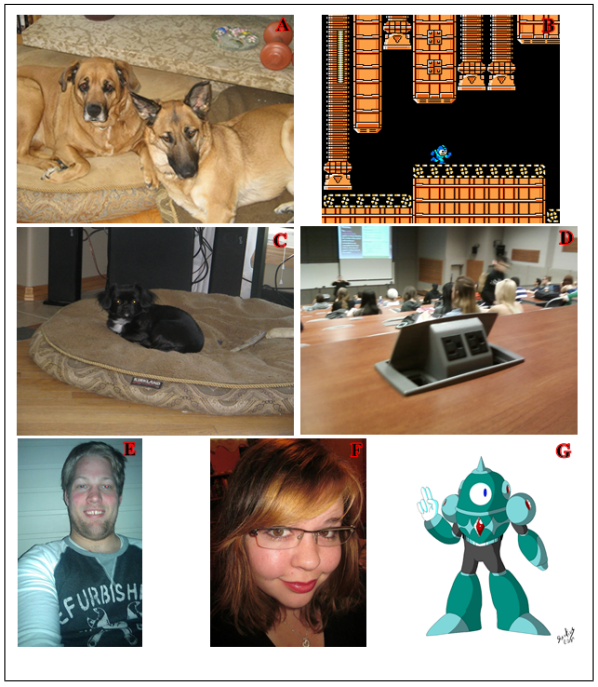



**Figure 1:** *Labelled images used for testing RCGIP software. A. Image with lots of similar colours. B. Digital video game image. C. Dark, low contrast photograph. D. Bright, high contrast photograph. E. Bright, low contrast photograph. F. Dark, high contrast photograph. G. Digital art with shading.*

2. Retrogressive Quantization:
   Contrast Stretching, Histogram Equalization, 2D DCT
3. Gamma adjustment (O) → Bilateral filtering (O) → Emphasize Edges (O)
4. Convert RGB to image to indexed image by median cuts [7, 3]
5. Convert indexed image back to RGB image

The two most crucial operations are the required retrogressive quantization, and conversion from RGB to an indexed image via median cuts. In particular, direct translation from the original RGB space to a reduced palette produces less than desirable results. Better results may be achieved in altering the spread of pixels in RGB space in such a fashion that visually pleasing matches are made to the target palette subject to median cuts. Greater detail on this point, and the utility of optional operations is presented in the section that follows.

## I. Experiments and RCGIP Stages

The following discusses the various individual RCGIP operations in greater detail with specific reference to experimental observations. In most cases, Intermediate processing steps may assist in achieving a final result that is more consistent with expectation, with all transformations occur before quantization to a particular colour palette. In addition, we comment further on operations that afford desirable output with reference to the properties of input images, and corresponding console system palette for each of these operations to facilitate general use of the proposed methods.

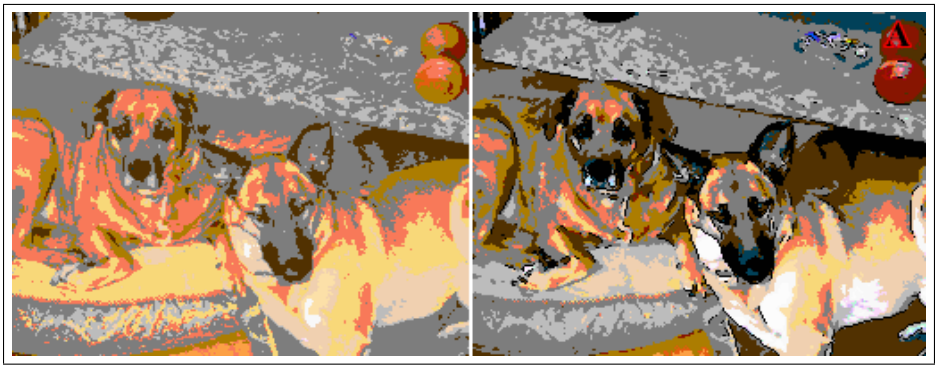


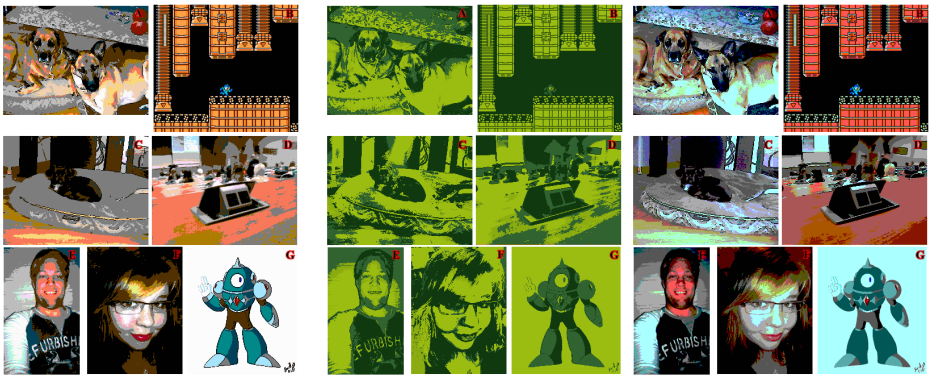
**Figure 2:** *Example image with retrogressive mapping via median cuts to NES palette and no* optional *processing (left), and contrast stretching and gamma adjustment (right).*

Tests were performed based on seven images chosen to span a broad set of properties, and distinct categories. These are depicted in Figure 1. Specific properties of the images used for testing are listed in the figure caption.

**Down-sampling** For older console games (NES, SMS, GB), downsampling the image (spatial quantization) has the impact of producing a result that is more consistent with the look of classic games (e.g. NES resolution is 256x240 pixels). This operation is important in producing high-quality retrogressed images by creating a pixelated representation at a resolution that approximates that of the NES. Images were sub-sampled to have a width of 256 pixels, while maintaining the aspect ratio of the original.

**Retrogressive Quantization** Perhaps the most paramount operation in the overall processing pipeline, is ensuring that original pixel values are spread in a fashion that results in a median-cut remapping of colour tones with properties that are complementary to the target output palette. Given unprocessed input images, there is generally insufficient contrast to produce a color-quantized result with appropriate contrast in the target palette space. Modifying the initial RGB values to have a broader spread ensures that the console palette is well used in representing contrast within the source image. In most cases, multiplicative contrast stretching with an offset (so that values reside in [0,1]) is sufficient to achieve this. An exception to this is the 2-bit GB palette; it is especially challenging to divide pixels into one of 4 intensity values, and retain a high quality representation of image content. To overcome this limitation, one can modify the image structurally by thresholding of local 2D DCT coefficients of the R, G and B channels. Applying a low-pass threshold to DCT coefficients filters out high-frequency variation and results in a higher level of quantization, and blocky appearance. Interestingly, this serves a similar role to the bilateral filter used to produce cartoon-like images in controlling detail preservation. In addition, given the nature of the DCT2 basis functions, the checkered appearance of DCT coefficients lends itself well to creating an overall look that is reminiscent of classic games. Despite the limited palette of the GB, comprised of only four intensities, results achieved for this case are surprisingly consistent with what is typical of games on this system.

Figure 2 shows an example of RCGIP mapping to the NES palette in the absence of any pre-processing (left). Dark yellow is observed, and many RGB values are mapped to common colours destroying boundary structure. We found experimentally that rescaling the channels to [0,1], and applying gamma adjustment ($\gamma = 1.2$) to darken the image improved the quality of output in terms of colors represented. Further enhancement based on the edge emphasizing operation is shown in Figure 3. Poor contrast in a source image can result in a lack of contrast when mapped to a more primitive palette. This demonstrates the importance of both artificially stretching the contrast



**(a)** *NES* **(b)** *GB* **(c)** *SMS*

**Figure 3:** *Retrogressive mapping of the test images to various console palletes with optimal choices for optional processing stages.*

(beyond a level that is visually appealing in RGB space), and altering RGB values for edges prior to median cuts color indexing.

**Gamma Adjustment** Although initial stretching of the palette by re-scaling each of the R, G and B bands, or histogram equalization ensures that the target console palette is well used. Overly dark, or bright images benefit from additional gamma adjustment (especially for contrast stretching). Gamma adjustment was found to be especially useful in instances where an image was predominantly dark, or predominantly bright in allowing detail to be retained in the compressed palette space. This control is also of importance in considering the target palette. In the case of the SMS palette, there is poor representation among brighter colors, and as such it can be useful to apply gamma adjustment to compresses darker intensities while expanding the distance between brighter intensities prior to re-mapping via median cuts. This is evident in examining Figure 3, where the lack of near–white values in this palette results in a mapping to bright-blue. That said, this is similar in style to some SMS console games, but may not be desirable for all cases.

**Bilateral Filtering** Bilateral Filtering [8] is a strategy that has been applied successfully towards image abstraction, or *cartoonization*. While the aim of the current work differs slightly, this is nevertheless a useful operation in the pipeline, towards allowing for control over the level of detail in the final result. In particular, in stretching the range of values in the RGB bands, it is possible to elicit unwanted noise in the final result. Based on experimentation, values of $W = 1$, and $\sigma = 1$ (bandwidth of Gaussian smoothing on spatial, and intensity values respectively) on the spatially sub-sampled representation were found to provide appropriate edge-preserving smoothing consistent with a classic-game look. This operation was found to be useful in employing highly quantized palettes in preserving smooth shading effects. While there is no particular instance where this operation is critical, it serves much the same purpose as its use in producing cartoon-like images in allowing further flexibility in abstracting the image to a desirable level.

**Edge Enhancement** Classic games often have highly emphasized edges not present in natural images. Edge enhancement may be applied to produce cartoon-like output [11] and was also found to provide output more consistent with the look of classic games. Edges were detected based on a 3x3 Sobel detector at the sub-sampled resolution, and subsequently subtracted from the processed image with values clamped at 0. This has the effect of producing dark boundaries following median-cuts quantization. This is reflected in the images shown in Figures 2 (right) and 3.

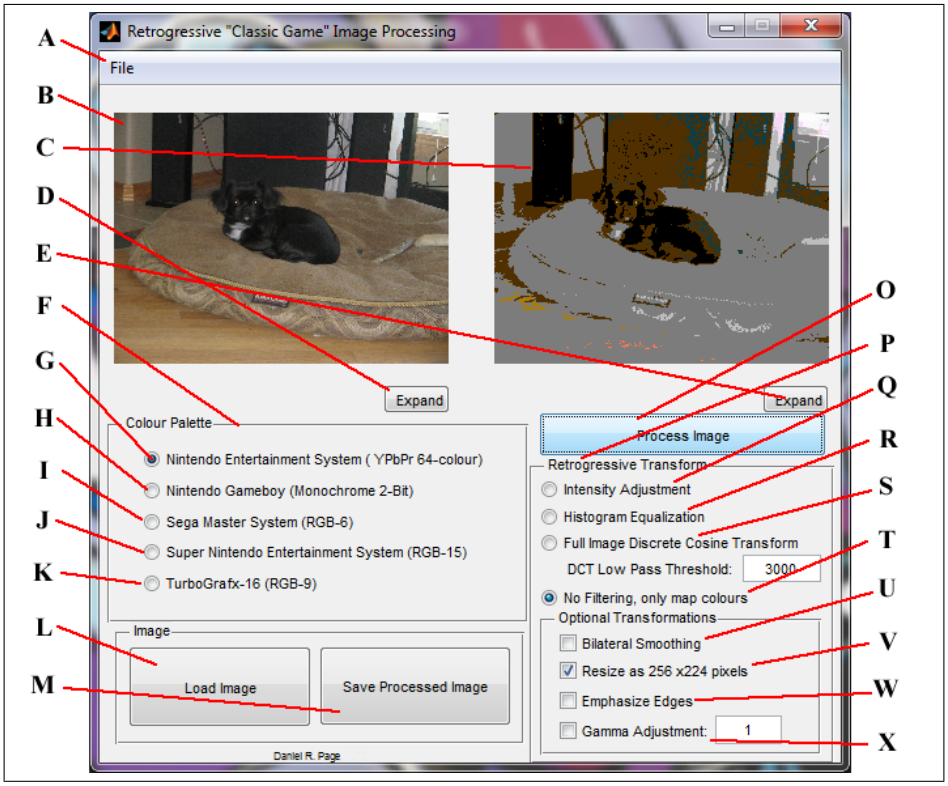



**Figure 4:** *Labeled screenshot of Retrogressive Classic Game Image Processing graphical user interface window*

## IV. RCGIP Software

The *Retrogressive "Classic Game" Image Processing* (RCGIP) software has been made available for general use[1]. The RCGIP software allows a user to load images, select from optional processing stages and quickly optimize output of the RCGIP pipeline to a particular colour map. Figure 4 provides a screenshot of the graphical user interface with a sample image loaded and processed with default settings.

### I. Overview of Software

The following describes operations that may be exercised within the software, in referring to the labels found in Figure 4. To begin, a user loads an image. This can be accomplished through the pull–down menu at (A), or the button labelled (L). When an image is loaded, it appears in the axes (B), and any retrogressed image output will appear to the right (C). The user may zoom in or out of either the input, or output by clicking the "Expand" button for each of D, or E respectively. The user can select a target palette in the Colour Palette panel located at (F). In the panel Retrogressive Transform (P), the user can select one option for retrogressive quantization (Q – S), and any number of the different optional transformations (U – X). The output mapping is produced in clicking "Process Image" (O). The user can modify settings and update until a desirable result is achieved. Finally, the user can save the output (M). This allows for quick iteration to the most desirable output

[1] http://www.mathworks.com/matlabcentral/fileexchange/39613-retrogressive-classic-game-image-processing/

for a particular image set, and target palette. Following this, individual operations corresponding to those selected may be applied offline to process a larger set of frames or artwork.

## V. Conclusions

In this paper we demonstrated an effective processing pipeline to create retrogressive classic images for a variety of types of input imagery. This is achieve through a processing pipeline that involves shifting native RGB values in colorspace via point processing or structural modification via the discrete cosine transform, with accompanying optional bilateral filtering and edge enhancement. The methods implemented and associated software can be useful to those who wish to adapt their images to emulate a classic game appearance. Discussion of the role of various stages of the processing pipeline, with reference to experimentation reveals a number of important observations which serve as a set of guidelines for quickly adapting content to achieve output for a desired target console palette.

## References

[1] Paul Debevec. A median cut algorithm for light probe sampling. In *ACM SIGGRAPH 2006 Courses*, page 6. ACM, 2006.

[2] Mark Guttenbrunner, Christoph Becker, and Andreas Rauber. Keeping the game alive: Evaluating strategies for the preservation of console video games. *International Journal of Digital Curation*, 5(1):64–90, 2010.

[3] Paul Heckbert. Color image quantization for frame buffer display. *ACM SIGGRAPH '82 Proceedings*, 1982.

[4] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460. ACM, 1998.

[5] Barbara J Meier. Painterly rendering for animation. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 477–484. ACM, 1996.

[6] Frank Palermo, James Hays, and Alexei A Efros. Dating historical color images. In *Computer Vision–ECCV 2012*, pages 499–512. Springer, 2012.

[7] Spencer W. Thomas. Efficient inverse color map computation. *Graphics Gems II, (ed. James Arvo), Academic Press: Boston*, 1991.

[8] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *In Proceedings of IEEE International Conference on Computer Vision*, 1998.

[9] Shijie J Wan, SK Michael Wong, and Przemyslaw Prusinkiewicz. An algorithm for multidimensional data clustering. *ACM Transactions on Mathematical Software (TOMS)*, 14(2):153–162, 1988.

[10] Meng Wang, Richang Hong, Xiao-Tong Yuan, Shuicheng Yan, and Tat-Seng Chua. Movie2comics: Towards a lively video content presentation. *Multimedia, IEEE Transactions on*, 14(3):858–870, 2012.

[11] Holger Winnemöller, Sven C Olsen, and Bruce Gooch. Real-time video abstraction. In *ACM Transactions On Graphics (TOG)*, volume 25, pages 1221–1226. ACM, 2006.