Framework for implementing, evaluating and analyzing some algorithms for non-photorealistic image processing

Framework zur Implementierung, Auswertung und Analyse von Algorithmen für nichtfotorealistische Bildverarbeitung

Virginia Todorova Dimitrova

Faculty of Computer Systems and Technologies, Technical University of Sofia
Sofia, Bulgaria, vergy@tu-sofia.bg

***Abstract* —** The article presents a desktop application that supplies a framework for designing, implementing, testing and evaluating different techniques for non-photorealistic image processing. The framework supplies a common template and an effective interface for programmers who want to implement and analyze a lot of non-photorealistic image processing algorithms. The goal was to allow users to investigate the applicability, functionality, effectiveness of already implemented algorithms for non-photorealistic image processing as well as to implement new versions of the same algorithms and new algorithms in the target area. The main focus of the realization was to develop an extendable system of closely interconnected modules that give the user full control over the total process of loading image, performing needed preprocessing steps, applying selected image processing technique, viewing and storing the resultant images, obtaining information concerning effectiveness of the applied algorithm and manage the history of the performed actions. In this stage the framework can be used for pixel-based non-photorealistic algorithms including such popular techniques as digital halftoning and screening.

***Zusammenfassung* —** Der Artikel präsentiert eine Desktop-Anwendung, die ein Framework für das Entwerfen, Implementieren und Vergleichen verschiedener Techniken für die nicht-fotorealistische Bildverarbeitung liefert. Das Framework bietet eine gemeinsame Vorlage und eine effektive Schnittstelle für Anwender, die viele nicht-fotorealistische Bildverarbeitungsalgorithmen implementieren und analysieren möchten. Ziel ist es, den Anwendern die Möglichkeit zu geben, die Anwendbarkeit, Funktionalität und Effektivität bereits implementierter Algorithmen für die nicht-fotorealistische Bildverarbeitung zu erforschen sowie neue Versionen derselben Algorithmen und neue Algorithmen im Zielbereich zu implementieren. Der Schwerpunkt der Realisierung liegt in der Entwicklung eines ausziehbaren Systems von eng miteinander verbundenen Modulen, die dem Anwender die volle Kontrolle über den gesamten Prozess des Ladens eines Bildes, die Durchführung der benötigten Vorverarbeitungsschritte, die Anwendung der ausgewählten Bildverarbeitungstechnik, das Betrachten und Speichern der Ausgabebilder, um Informationen über die Wirksamkeit des angewandten Algorithmus zu erhalten und die Geschichte der durchgeführten Aktionen zu verwalten. In diesem Stadium können das Framework für pixelbasierte nicht-fotorealistische Algorithmen sowie solcher populären Techniken, wie digitales Raster und Screening, verwendet werden.

I. Introduction

Images are fast becoming a generic data type for general-purpose computer systems and thus a need to process a huge amount of digital image data appears. One of the goals of this type of image processing can be formulated as creating imagery that is effective at conveying information, expressive and beautiful. Image processing algorithms that lay in the core of non-photorealistic rendering (NPR) have the potential to achieve this goal through generating by computer and by user demand images that are effective, attractive and appropriate in the context of the human-computer dialog. These algorithms once implemented will allow to generate meaningful imagery for end users thus enabling them to observe information in graphical form. An overview of NPR techniques can be found in the textbooks by Strothotte and Schlechtweg [1], Gooch and Gooch [2].

A lot of algorithms in image processing are well known from decades and effectively implemented as a build-in functionality of popular program packages like Photoshop, Photoscape, GIMP. Groups of internally connected algorithms are implemented as plug-ins for such program packages. A huge variety of image-processing algorithms are exposed as library functions that can be invoked from applications developed in different programming languages, for different platforms as well as for a lot of devices (desktop, laptop, tablet, mobile devices). The goal of the present work is completely different in at least 2 aspects. First, most of image processing algorithms are implemented in such a way to reduce the artifacts in images induced as a result of the applied image-processing techniques. The present work adopts the same algorithms but slightly modified to induce image artifacts allowing the achievement of various visual effects typical for NPR Second, the above mentioned implementations of algorithms (as build-in functionality, plug-ins, libraries of

functions) allow users to see the results of different image-processing techniques applied to a lot of images. However, these implementations do not allow the user to modify a particular algorithm, to easy add new implementations of the existing or new algorithms, to analyze and evaluate the effectiveness of different implementations.

So, the goal of the presented work is to develop a program solution that serves as a framework for implementation, evaluation and analyzing of selected algorithms in NPR. The user has the ability to use a known template and a common user interface in the process of implementing, evaluating and analyzing the algorithms.

The present work covers a useful area of non-photorealistic image processing called halftoning as well as a special kind of halftoning called screening. There are different techniques for halftoning like patterning, ordered dithering, error diffusion and for each technique there is a variety of algorithms for its realization. The common feature of these algorithms is that they encode the "subjective" artifacts into the resulting image that clearly do not exist in the original image.

All implemented NPR techniques work on the basis of altering images that encode intensities on a per-pixel basis. Intensities are created either by a photorealistic renderer or in a real photographic process. The image is interpreted simply as a pixel matrix, where each entry is either a gray level or a color value (RGB). The pixels are manipulated to produce various different visual effects.

All discussed algorithms require an input image in Bitmap format. Some of the algorithms need no more information than that directly accessible from an input image. Others require a user input in the form of numerical parameters, short text, or reference images. These additional parameters are specific for each class of algorithms and are very important, because the obtained visual effects strongly depend on the values of these parameters. The resultant non-photorealistic images with desired visual effects are generated semi-automatically by analyzing input images, extracting certain information, and accepting a lot of user input.

The main contributions of this paper are two: support a complete programming tool for implementing, evaluating and analyzing some NPR techniques; and demonstrate with a set of already implemented algorithms the possibilities of these techniques. During the program realization are created both: 1) a library of already implemented algorithms; and 2) a couple of modules that allow to implement additional algorithms with no need to think about some common steps like loading images, presenting images, logging the performed actions.

The specific contributions of this paper are:

- An extendable library of algorithms implemented in C#;
- A common template and a common user interface;
- Interfaces for an interactive design of patterns, dither matrices and artistic screens.

II. Related Work

Advances in the field of NPR show the usefulness of image-space approaches. An important class of NPR rendering styles operates directly in image space, such as halftoning, screening, stippling, or hatching methods. In the field of NPR there are two branches: techniques for producing NPRendering from 3D models [3, 4] and techniques for applying NPR styles to 2D images. Typically graphics packages for 3D modeling like 3DS MAX supply tools that allow to use image space techniques to post-process renderings of 3D scenes in order to obtain a lot of non-photorealistic styles. For example, 3DS MAX exposes a special kind of material named Ink'n Paint that can be applied to one or more objects of a 3D scene to render them in a cartoon style. Additionally, the user has the possibility to view the created 3D scene in a lot of non-photorealistic styles. NPR algorithms implemented in 3ds max usually use 2 ½ D (G-buffers) and 3D data as an information source. Some image-processing algorithms in 3ds Max are applied to the content of a variety of G-buffers generated as a co-product of the obvious rendering process [5].

The second branch of NPR encompasses algorithms for artistic drawing styles that work directly in image space. A plenty of such algorithms are implemented in commercial programs for image processing like Photoshop. Photoshop offers a lot of filters that allow turning a photo into a painting. These filters are grouped in different categories (Artistic Filters, Texture filters, Stylized filters and many others) and replicate natural or traditional media effects or simulate the effect of halftone screen. Some NPR techniques for image halftoning such as ordered dithering and error diffusion are widespread in programs for image processing, which are desktop and web applications like MATLAB, Ximagic GrayDither for Photoshop, Meemo. MATLAB provides Bayer dithering [7, 8] by supplying dither() function which converts the grayscale input image into the binary image. The web application "Meemo" supplies Akiston, Bayer and Floyd-Steinberg algorithms for automatically transforming an imported image into a binary image [9]. Ximagic GrayDither is a Photoshop plugin that allows to reduce color/grayscale images to an n-level grayscale using dithering [10]. For Windows platform a lot of image filters some of which non-photorealistic (like Image Distortion Blur Filter) are implemented as separate applications and included in MSDN [11]. All of them supply a lot of parameters that the user can set to change slightly the obtained visual effects.

In the area of NPR there are also a lot of software tools supplied in a form of software development kits like NPR Kit for MODO and REDsdk. The NPR Kit for MODO supports the creation of a variety of familiar artistic effects in MODO including 'toon' shading, stippling, halftones and edge rendering. REDsdk is a software tool that supplies two styles of rendering: Cartoon rendering and Scetch rendering, and can be used by programmers to generate NPR images [12]. Some image-space algorithms that can be classified as non photo-realistic are implemented as classes that make them easy accessible for programmers (EffectFactory class in Android is such a class) [13].

III. Structure and Scope

Overall structure of the presented work is shown in fig. 1. To give a sense of the system from users' viewpoint the modules structure and the organization of the user interface are briefly described.

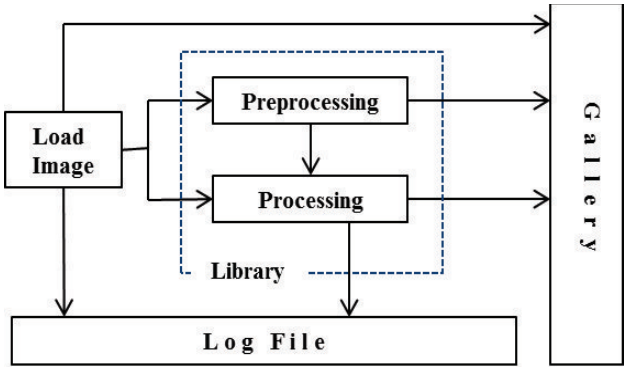

Fig. 1. The main structure of the presented work.

One of the main modules is a library of already implemented NPR techniques (Library). The Library was developed as a separate project using ClassLibrary template of Visual Studio .NET. A reference to the Library was included into the application and all implemented algorithms are accessible through the Preprocessing and Processing top-level menu commands. Each non-photorealistic image-processing algorithm is implemented as a separate method. Methods that perform similar types of NPR techniques pertain to a separate class. This is the approach applied by the author to categorize the included NPR techniques. The name of the class and the name of the method form a fully qualified name used to specify and subsequently identify the corresponding NPR technique. When a new NPR technique has to be implemented the selection of the appropriate class which includes the definition of the method is the responsibility of the programmer.

Another module is the module for loading input images for processing from disk files in one of the supported formats (BMP, GIF, EXIG, JPG, PNG and TIFF). During the loading of the input image metadata form the image file is extracted, a new object of ImageProperties class is created and its fields are partially populated on the base of the extracted metadata. The object of ImageProperties class is stored in the collection of such objects and a corresponding record in the Log file is added. Another operations performed by this module include: creating a Bitmap object that consists of the pixel data for a graphics image and its attributes; and resizing the picture box control in order to prevent interpolation of the input image.

Since one of the main goals of the present work is to allow the user to evaluate and analyze different NPR techniques, the framework supplies two important modules. The first one is the Log file, created and maintained automatically by the application. This Log file is created and opened for writing each time a new input image is loaded for processing. Then an information about the loaded input image and about all actions performed over it is regularly recorded into the file. The collected information is stored as a text file with separate paragraphs each beginning with a predefined keyword (Image, Technique, Elapsed Time). The content and the structure of the Log file are designed to be used for analytical evaluation as well as for subsequent analyzing of the implemented and applied NPR techniques. The user can compare the parameters of the input image and the output images; trace the applied preprocessing and processing steps and see the elapsed time for each applied NPR technique by reading the paragraphs in a log file. It is possible to view the content of the log file at all times during the current image processing session.

The last module is the so called Gallery dedicated mainly for visual evaluation of the obtained results in a particular image processing session. The Gallery is a separate window that shows the input image as well as all output images obtained as a result of applied preprocessing and image processing techniques. Each image is represented in the Gallery as a thumbnail image in other to obtain a common view over the results of the particular image processing session. The user however has the possibility to see each included image in real dimensions. There are two options for each represented in the gallery output image: to obtain detail information concerning the image and to delete the image. Additionally, because the zoomed part of each processed image is of great importance for evaluation, the user has the possibility to interactively select a pixels' region and zoom it thus making the induced artifacts more visible and available for examining.

The main focus in the realization is on achieving the high level of interactivity. So, all user actions from loading an input image to presenting the output images are performed interactively. Conventional menus and dialog boxes are used when needed. The main challenge is on preparing the additional source data consisting either of simple numerical and textual input, or of more complex pixel patterns, dither matrixes, threshold matrixes and dither screens. First, this source data has to be prepared in such a way to induce artifacts in original images to achieve various visual effects. Second, the user must have the possibility to dynamically and easily change this source data. In order to achieve the needed level of interactivity a separate custom dialog boxes are supplied for preparing pixel patterns, for preparing dither matrices and for histogram equalization.

IV. Details of Realization

Most image files contain metadata that allows to determine features of the image such as the number of bits per pixel, number of pixels in each row, and number of rows in the array, a color table. This metadata can be read programmatically, maintained in the memory during the image processing and saved in the output file. One of the ways to save information about the applied image processing techniques over the content of the loaded input image is to use the metadata stored in an image file using the predefined PropertyItem class and the Image.PropertyItems property. GDI+ stores an individual piece of metadata in a PropertyItem object. This object can be used both to read existing metadata (GetPropertyItem method) and to write new metadata to image files (SetPropertyItem). But a PropertyItem object has only four properties: Id, Value, Len, and Type that are not enough for the goals of the present work. Additionally, it is difficult to set property items using these predefined classes. So, the author adopts another approach to manage and store metadata – create a custom class ImageProperties. All objects of ImageProperties class are stored in a separate collection in the memory. This information is the corner element of the Log file and is used to explain details about the selected image in the Gallery.

One of the focuses in the realization is on storing, managing and representing the results. During each session of image processing a lot of output images are created. For each input image as well as for all generated output images a corresponding thumbnail image is created. Both the input image and the set of output images are stored in memory for the time of the image processing session in two separate collections: one collection of images with original size and the other – with their thumbnail versions. This approach allows the user to see by demand the Gallery of all images he/she works with in the current session represented as thumbnails as well as to see each selected image of the Gallery in real dimensions.

V. Functionality

A. Preprocessing

Some of the implemented NPR techniques require one or more preprocessing steps. These preprocessing steps include: conversion of a colored image into a grayscale image; conversion of a grayscale image into a binary image; creating an image histogram; performing histogram equalization. There are a lot of algorithms for conversion of a colored image to a grayscale image (grayscale filters like "single color channel", "averaging", "luminance", "desaturation", "de-composition" and so on [14]). The presented work supplies 4 such algorithms among which the user can make his/her choice. The simplest method for converting a grayscale image to a binary image is by thresholding, i.e. a two-level quantization. The implemented algorithm allows simple quantization using a fixed threshold in the quantization process. Preprocessing is used also for preparing a dither screen with needed qualities for the

implemented screening NPR technique. For this purpose the framework exposes two methods: to compute a histogram of an input image that will be used as a threshold matrix; and to perform a histogram equalization in order to achieve a uniform distribution of the gray values.

B. Processing

The first implemented halftoning technique is a patterning technique. The patterning algorithm is implemented for a bilevel system and a rectangular grid of n by n pixels, allowing to produce n+1 intensity levels. The user has the possibility to specify the size n of the rectangular grid of pixels (pixel-grid size). Pixel patterns for any pixel-grid size are specified with a "mask" of pixel position numbers. The user has the possibility to populate the mask interactively and to see the generated pixel patterns.

Next implemented halftoning technique is the error diffusion technique that utilizes the result of a simple threshold quantization and redistributes the resulting error to the surrounding region of a pixel. Generally, there is a wide range of algorithms for error diffusion. Only the basic and most popular error diffusion algorithm proposed by Floyd and Steinberg [15] is implemented in the Library. Implementations of other algorithms for error diffusion are left to the users of the framework.

The slightly modified Floyd-Steinberg algorithm is used for implementation of a specific halftoning technique - halftoning using lines. This separate NPR technique makes use of error diffusion to get an initial pixel distribution which is then changed to introduce image artifacts explicitly. Artifacts are introduced by drawing short lines that cover several pixels instead of single dots in the output image.

The next implemented halftoning technique is ordered dithering technique. The emphasis of the implementation is on the designing of dither matrices in which pixels are grouped in such a way to form visual patterns to be introduced in the image for achieving interesting effects. A dither matrix was defined by a so called index matrix that is a special case of a family of dither matrices first defined by Bayer [16]. The user has the possibility to both interactively enter the size of the dither matrix and populate its elements and thus to investigate how the size and the content of the dither matrix influence the quantity and the quality of the induced artifacts.

The second implemented NPR technique is screening interpreted as a separate halftoning technique with which two images - an original image and a dither screen, are combined using a special algorithm. The user has the possibility to use an arbitrary image as a dither screen but because there are special requirements for uniform distribution of threshold values and for homogeneous spatial distribution of threshold values the chosen image must be preprocessed.

At last, a special screening technique called screening with text is included. In this NPR technique letter shapes or texts are used as dither screens. The resultant images look like normal digital halftoning from a distance, but when viewed from close up they reveal the text being brought into the rendition by the dither screen used.

VI. Conclusion and Future Work

The results of the present work can be used for studying different NPR techniques with coding, for evaluation of the obtained non-photorealistic effects and for analyzing the program implementations. The main focus of the realization is to develop an extendable system of closely interconnected modules that give the programmer full control over the total process of loading image, performing needed preprocessing steps, applying selected image processing technique, viewing and storing the resultant images (visual evaluation), obtaining information concerning effectiveness of the applied algorithm (formal evaluation) and manage the history of the performed actions (analyze the results). The framework also provides a limited control over the already implemented algorithms through parameters and additional source data specified with a high level of interactivity.

In this stage the framework can be used for pixel-based non-photorealistic algorithms including such popular techniques as digital halftoning and screening. The implemented NPR techniques work over the content of images obtained through scanning, digital photography, or photo-realistic renderer. All algorithms operate on the level of pixels and have one single goal - to transform the intensity of the corresponding pixels in such a way to achieve various visual effects.

There are three main aspects for future work. The first is to extend already implemented screening with text technique in order to allow constructing dither screens from arbitrary contours (contour-based screening). The second is to implement such popular image-space NPR techniques as stippling and image mosaics. The third aspect is to extend the scope of the implemented NPR techniques (working in 2D image space) by adding algorithms that operate over the content of G-buffers (2 ½ D sources of data).

References

[1] T. Strothotte,S. Schlechtweg, "*Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*" Morgan Kaufmann, 2002.

[2] B. Gooch and A. Gooch, "*Non-Photorealistic Rendering*" AK Peters Ltd, 2001.

[3] M. Eißele, D.Weiskopf, T.Ert, , "The G2-Buffer Framework" Institute of Visualization and Interactive Systems, University of Stuttgart unpublished.

[4] T. Saito and T. Takahashi, "Comprehensible Rendering of 3-D Shapes" Computer Graphics (Proceedings of ACM SIGGRAPH 90),volume 24, p. 197–206, 1990

[5] V. Dimitrova, "Obtaining G-buffers for Non-Photorealistic Rendering: Software Tools And Procedure", *Computer& Communications Engineering*, 2015, vol. 9, Technical University of Sofia

[6] A.Hertzmann, "*Algorithms for Rendering in Artistic Styles*" New York University, May 2011

[7] Alec Jacobson, "Dithering in MATLAB", http://www.alecjacobson.com/weblog/?p=1624

[8] MathWorks, MATLAB official web site, https://www.mathworks.com/products/matlab.html

[9] Meemoo application official web site, https://app.meemoo.org/#gist/3721129

[10] Ximagic Photoshop plugins web site, http://www.ximagic.com

[11] https://code.msdn.microsoft.com/windowsapps/Image-Distortion-Blur-673b1bdd

[12] Non photo-realistic rendering with REDsdk, http://www.redway3d.com/downloads/public/documentation/bk_re_non_photo_realistic_rendering.html

[13] Android: Image Processing, https://developer.android.com/reference/android/media/effect/EffectFactory.html

[14] Seven grayscale conversion algorithms (with pseudocode and VB6 source code), Tanner Helland, Oct 1, 2011, http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

[15] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale*", Journal of the Society for Information Display*, vol. 17, no. 2, pp. 75–77, 1976

[16] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures" IEEE International Conference on Communications, vol. 1, June 11-13 1973, pp. 11–15.