Non-Photorealistic rendering: pencil drawings effect

Luca Buratti
University of Illinois at Chicago
luca.buratt@gmail.com

Giorgio Conte
University of Illinois at Chicago
gconte2@uic.edu

Walter Dworak
University of Illinois at Chicago
wdwora2@uic.edu

ABSTRACT
In this article we are going to present and discuss a simple and effective approach to render pencil drawing effect. We introduce a couple of methods, one that render the image in gray-scaled colors, while the other that also render colors. The main idea was taken from two articles, but on that basis we came up with some new ideas that improve the approach described. Thanks, to our ideas it is possible to render in a better way the colors of the image filtered, and at the same time we can reach a more blurred effect. Thus, the general user perception is than improved and with our filters it is possible to render different interesting NPR effects.

Author Keywords
non-photorealistic rendering; pencil drawings; pencil effect.

INTRODUCTION

Pencil drawing is one of the most important pictorial languages in the image arts to abstract human perception of images and natural scenes in general. The artistic style is characterized by a soft and hazy appearance meant to invoke a sense of calm in the viewer [11]. So far, the techniques proposed to achieve this goal can be subdivided into two categories: 2D image-based rendering and 3D model-based rendering. The first techniques proposed to obtain this kind of effect were focused on 3D models. However, the increasing popularity of digital cameras in the last decade has increased the demand for rendering pencil drawing from natural images more and more. The literature details some different techniques for model and image-based pencil style NPR effects.

RELATED WORK

As stated in the introduction, the most important approaches present in the literature can be classified into two groups: model-based and image-based approaches. These must deal with a number of factors to generate a convincing pencil effect: drawing direction, color blending, and paper texture. This section will describe the major techniques found in the literature according to the classification proposed.

Model-space algorithm

Achieving a pencil drawing style effect using rendering techniques on 3D model scenes is a known approach in the literature. There have been very good results in this field since 1999. For instance, the tool for pencil drawing proposed by Sousa and Buchanan [9] achieved high quality results simulating the different characteristic of graphite pencils on various paper types. However, real-time rendering of 3D meshes in the pencil drawing style is a difficult goal to achieve. Rendering and projecting 3D meshes and then apply 2D image-based filters can be used in order to achieve pencil drawing effects but performances may not be enough for animations. In 2002, Webb proposed a real-time hatching technique [12] which can stroke textures onto a 3D surface, this work shows that is possible too achieve the required perfomances for real time in this field. In 2006, Lee proposed an effective technique [6] that was able to generate remarkable results in real time using interior shading with hatching and multiple contour drawing and presenting techniques for generating and mapping pencil textures that reflect characteristics of graphite pencils on paper.

Image-space algorithm

Many image-based techniques have been proposed and they exploit several different methods and filters, and among these techniques more refined classifications can be done. In fact, some techniques depend only on one filter while others are a more complex combination of filters computed from the original image. The other characteristic that makes the techniques different is the color rendering. The vast majority of the methods, presented in the literature, create pencil drawings in a gray-scale format and only a few of them succeed in rendering colors with a pencil effect. Much research exists to solve singular issues with the generation of a correct color pencil drawing effect and only a few comprehensive methods.
One of the methods that is built upon to create the pencil effect specifically deals with the problems of line extraction and region smoothing on color images[3]. This effect, which out the paper texture simulation or pencil strokes, produces a cartoon-like image from the original. An edge flow field is computed from the input image and another filter finds edges in the edge flow field and simplify the edges as well as increasing the weight of major edges. The flow image is used again to do region smoothing. Similar areas are smoothed by decreasing the variation in the area's RGB values while maintaining the object shape in the image. The line image is then placed on top of the smoothed color image to produce a cartoon outline line effect from the original input.
One of the first methods to render a pencil effect was pro-

posed by Sun et al. [10]. In this work, the authors suggest an extension of Linear Integral Convolution (LIC). LIC was firstly introduced in [1] to visualize and render fluid motion such as the wind in a tornado. In particular, the result of LIC-based techniques is mostly dependent on, and it is biased by, the result of the white noises and the texture directions. Thus, they have presented a new way to produce white noises and texture directions so that the drawing quality is enhanced and it is closer to the real artistic style. With this technique the colors of the original image are not rendered, so the final image is represented in a gray-scale color. The LIC technique was expanded on to produce color pencil images[11]. The lines produced, however, appear rigid and unnatural. A region-based version of LIC exists which smooths out these lines.

In the same trend of the previous work, another convolution-based technique was advanced in [13], where the authors presented a 3-filter-based techniques. In fact, the input image is processed in parallel with three different filters and, in the end, the images are fused together to obtain the final output. The three tasks computed on the initial image are: a pencil filter, a black noise filter and an edge detection filter. The first filter is in charge of creating a pencil texture. To achieve this goal, they assume that the graphite marks present stochastic distribution according to the coarseness of papers and that the pencil stroke can be defined with three parameters: length, angle and width. The second filter create a black a noise starting from the initial image and they exploit the one presented by Mao in [8]. The last filter applied is the edge detection filters. Among the all methods available in the literature, the authors apply the one presented in [2]. The first two effect are convoluted and then the third is fused together. Also in this case the output image is rendered using gray-scaled colors.

Later a new modus operandi was delineated in [7]. This approach combines tone and stroke structures leading to a visually consistent pencil effect. Their strategy consists of two main steps: pencil stroke generation and pencil tone drawing. In fact, those two effects complement each other so that the stroke drawing gives an idea about the general structure of the scene depicted, while the tone drawing concentrates on shapes and shadow with the use of lines. Both of the two tasks are subdivided in two more sub-steps and then the two images are put together. This approach has been used mostly to produce gray-scaled images, but the authors show that the same method could be used to produce colored images.

One of the latest technique is proposed by Kim et al. in [4]. Their approach is based on the maximum filter and the images produced express edges and texture as an artist draws sketches with pencil. More in the details, the filter is composed by two step: first a sketch filter is applied to the image, then an edge feature extraction is used. Also this filter can be used to render colored or gray-scaled images.

Another technique looked at deals with the generation of a colored pencil drawing effect from a 2D image on a mobile platform. This method is intended to be computationally simpler than the other methods since it must be done on a lower resource device in a reasonable amount of time and still produce an accurate rendering of the effect[5]. An edge filtered sketch image is created from the original which will represent smooth areas as places to lighten and complex areas, edges, as places to darken the images. Smooth areas are assumed to be in a narrow range of intensity in comparison to their complex counterparts as well. The color filter, which this technique uses, separates regions based on their intensity ranges. This filter assumes the opposite of the edge based filter in that smooth areas are distributed across a wipe range of intensity levels. This color filter is applied to each RGB component separately and combined with a cumulative distribution function to generate the effect of an artists pencil.

OUR APPROACH

To render pencil drawing effect, we based our research starting from the techniques described in [4] and [5]. They introduced the so called "sketch filters" defined as follows:

$$s(x,y) = M \cdot \frac{f(x,y)}{m(x,y)} \tag{1}$$

where *M* is the maximum value of the intensity level. Usually it is 255, but in our case this value is equal to one. *f(x,y)* is the actual color of the pixel we are rendering, while *m(x,y)* is the maximum filtered image that is computed as follows:

$$m(x,y) = \max_{(u,v)\in w(x,y)} f(u,v) \tag{2}$$

w(x,y) is the window mask that is centered on the coordinates x,y. Then, we also slightly modified the sketch filter presented here, because the results were not very effective. The overall image produced tends to the be too bright and appear washed out. To reduce this negative behavior, we added a parameter, δ, that is summed to the function m(x,y). So, the modified sketch filter is computed as follows:

$$s(x,y) = M \cdot \frac{f(x,y)}{m(x,y) + \delta} \tag{3}$$

By tuning the parameter δ, we are able to reduce the brightness effect on the pixels. In fact, when the pixel intensity is a local maximum of the window it automatically becomes white, since the intensity is divided by itself. Thus, adding a constant value to the maximum, even though the pixel is the local maximum, the intensity value is not going to be one. So, thanks to this parameter, we can adjust the color level of the image. By increasing this value, the image produced tend to be closer to the real one, so it must be tuned wisely to obtain a "realistic" pencil drawing effect.

Moreover, we also introduced the mapping function that is computed as follows.

$$g(i) = \begin{cases} 0, & \text{if } x \leq \alpha. \\ M \cdot \frac{i-\alpha}{M-\alpha}, & \alpha < i \leq M \end{cases} \tag{4}$$

where *M* is the maximum intensity value that is, in our case, one and α is a free parameter that could be tuned manually. Basically, α can be considered as the boundary level between a smooth area and edge area. By applying this kind effect, it is possible to add more contrast in the image, giving more importance to edges and to the color in general. We apply this filter only when we are rendering colored images.

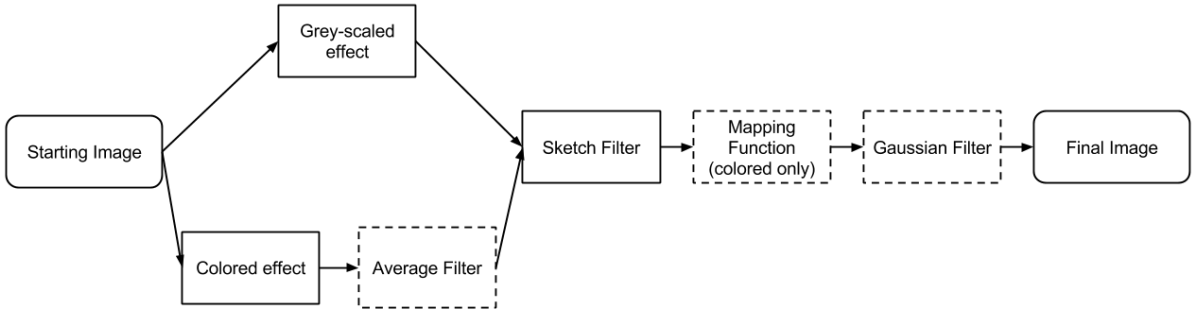

Figure 1. This figure shows the pipeline of the filter that we apply to the image. Filter in a dashed box are optional and could be activated/deactivated independently.

In the following subsections, we are going to present how we use these simple and basic concepts to render colored and gray scaled images.

Gray-scaled effect

To render gray scaled images, we needed to find a function that takes as an input the rgb values of a pixel and outputs the intensity level of the pixel. There are many approaches to do that and we tried a couple of them. The first function we used was a simple average of the three rgb values. Namely it was computed as follows:

$$Intensity(x,y) = \frac{red(x,y)+green(x,y)+blue(x,y)}{3} \quad (5)$$

However, this approach did not lead us to interesting results so we moved to a more complex formula. Thus, in our next approach the intensity of a pixel is computed as follows:

$$\begin{aligned} &Intensity(x,y) \\ &= \sqrt[2]{0.299 \cdot red(x,y)^2 + 0.587 \cdot green(x,y)^2} \\ &+ \quad 0.114 \cdot blue(x,y)^2 \end{aligned} \quad (6)$$

To render the scene, we build a window using the intensity values of the pixels and then we apply the sketch filter presented before to determine the gray scaled color of the pixel.

Colored pencil filter

For what concerns the colored pencil filter, we act in a similar way as before. However, conversely from the gray scaled image we do not need to sum up the rgb values into one intensity value, because we apply the sketch filter to each color channel. It is here that our modification gives the most effective results. However, applying the sketch filter to each color channel does not lead always to good results The colors obtained tend to be too bright or white completely. Thanks to the modified version of the sketch filter we are able to retrieve more contrast in the colors reaching much better results. We are going to discuss them more in detail in the next section.

Colored effect with mapping function

As we reported before, in our filters we also add a mapping function. Its aim is to introduce more contrast in the entire picture so that the edges and the shapes of the object inside the picture are more evident. In fact, some artists, when drawing with pencil, emphasize the shapes of the objects by drawing the border more than once. This filter is applied after the sketch filter so the result is the combination of the two. The free parameter α can be used to change the contrast. The higher is α value the more contrast there is in the image. From our experiments, in order to have a good pencil drawing effect this parameter should be not too high or close to zero. It is also important to note that $\alpha \in [0, 1]$.

Average Filter

We also decided to try a different approach. The idea is to create more uniform colors in the image by adding at the semi-difference between the color itself and the average color of the entire window mask to the color of a single pixel. So, the pixel color is computed as follows:

$$a(x,y) = f(x,y) + (f(x,y) - \frac{avg_{mask}}{2}) \quad (7)$$

Where $f(x,y)$ is the original color of the pixel and avg_{mask} is the mean of the colors in the pixel mask. Computed as:

$$avg_{mask} = \frac{1}{|w|} \sum_{(u,v)\in w(x,y)} f(u,v) \quad (8)$$

where $|w|$ is the number of the pixel contained in the window mask w(x,y). This effect by itself does not create notable results. However, by applying the sketch filter and the mapping function described above on the outcome of this function will generate an effective filter that tends to create more uniform

colors with a slightly blurry effect. This filter also works only with the colored images.

Gaussian Filter

The last filter that we apply is the Gaussian filter in order to obtain a blurred effect. Doing it this way, it is possible to give to the picture the blurred effect that it is usually present in pencil drawings.
The Gaussian filter is applied in two steps in order to be as efficient as possible. During the first step we apply the filter horizontally while in the second vertical stripes are considered. The number of pixels we take in account during this is equal to seven and the Gaussian curve that we use is centered in the origin and has $\sigma^2 = 0.8$. This configuration has been tuned experimentally in order to obtain, in our opinion, the best result. This filter is not customizable on the fly at this stage of our work, but it should be changed in the code.

Figure 1 shows the "pipeline" of the filters that we can use. This configuration leads us to create that many combinations of filters we have described so far.

RESULTS

This section is going to present and critically discuss the results we obtained. In particular, we are going to present different images obtained with different combination of filters and combining different values of the free parameters we have left in our implementation.

Sketch filter

The results of the sketch filters are showed in pictures 5, 6, 7, 8. Thanks to the extra parameter we have added it is possible to not lose the original colors and make the image more similar to a pencil-based drawing. The higher δ is the more realistic is the image, so usually this value should be very low in order to obtain a good pencil drawing effect. As it is possible to see from the images, thanks to this parameter we can reduce brightness of the pixels in the image that are creating a noise effect. However, the best results obtained by tuning this parameter can be seen in the colored version. In fact, when using the basic version of the sketch filter ($\delta = 0$) we lose the vast majority of the colors, leading to bad results. The correction we have applied fix and limit this bad behavior.

Mapping Function

As it is possible to see from figures 9 and 10, adding the mapping function does not drastically change the output. However, thanks to this filter it is possible to increase the contrast that is very low after the sketch filter. This light effect creates a much more defined image without ruining the pencil effect. Thanks to the parameter α, it is possible to tune the level of contrast in the output image.

Gaussian Filter

As it is known in academic literature, the Gaussian filter adds a blurry effect to the final image. The results can be seen in images 11, 12, 13. In these figures, the Gaussian filter has been combined with the different filters of the pipeline. The blurry effect can add more realism to the pencil drawing effect. As we were saying previously, the Gaussian curve we used is $\mathcal{N}(\mu = 0, \sigma^2 = 0.8)$ and the number of pixel taken in account is 7. The Gaussian filter should be tuned carefully, because the higher the variance is the more confusing the images are. This effect can render the blurred effect the pencil images have and the best results can be obtained with the black and white version of the filters. In fact, by applying this filter we are able to emulate the graphite spread in the drawing. This is a very used technique in black and white pencil drawing.

Figure 2. Chicago

Figure 3. Island

Figure 4. Mountain

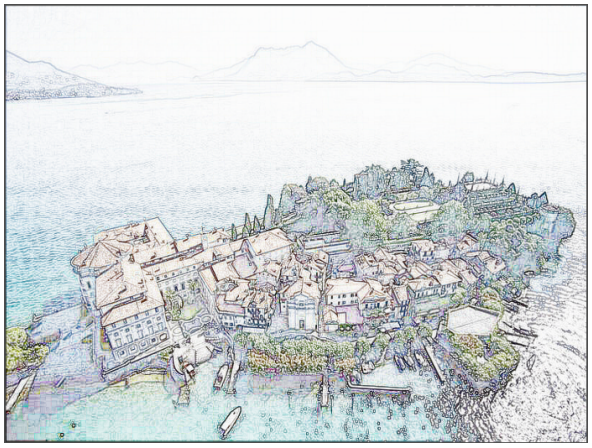

Figure 5. Sketch colored filter $\delta = 0.1$

Figure 8. Sketch black and white filter $\delta = 0.05$

Figure 6. Sketch colored filter $\delta = 0.05$

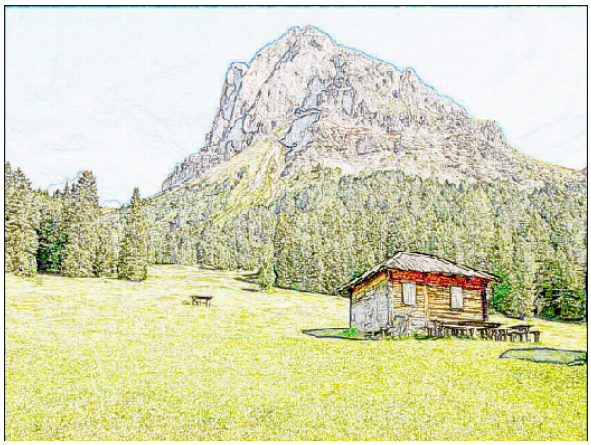

Figure 9. Sketch filter and mapping function $\alpha = 0.20$**.**

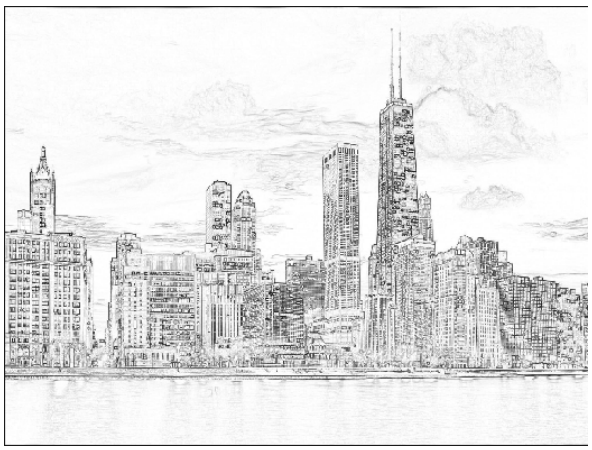

Figure 7. Sketch black and white filter $\delta = 0$

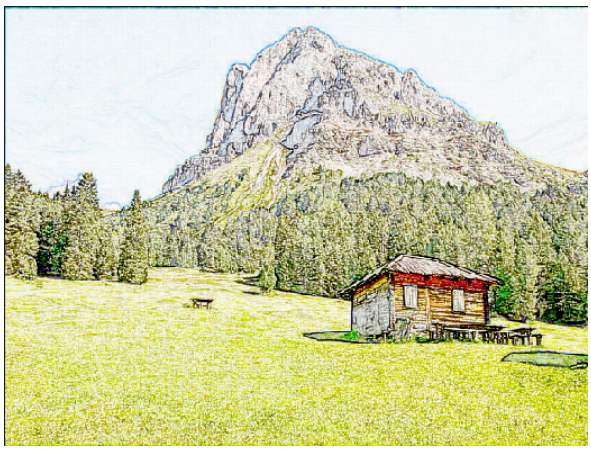

Figure 10. Sketch filter and mapping function $\alpha = 0.35$**.**

Figure 11. Gaussian Filter and sketch colored filter.

Figure 12. Gaussian filter, sketch colored filter and mapping function $\alpha = 0.3$

Figure 13. Gaussian filter, sketch black and white filter.

CONCLUSIONS

The filters described so far offer different kind of pencil drawing effects. We based our work on the the sketch filter and then we implemented and modified other filters in order to obtain better and more complete results. The final images are good, however there are still some problems. For example, when the input image present an homogeneous dark area the sketch filter transforms that area into a bright white and create an unpleasant effect on the overall image. Thanks to the introduction of the δ factor and the Gaussian filter, we are able to reduce and limit this bad behavior. Nevertheless, this problem is not deleted completely.

Still, the results obtained are good even though there could be room for improvements. A more effective technique to render pencil stroke could be added to this work. Thanks to the filter pipeline and to the many different free parameters we have left and it is possible to create many divergent and creative non-photorealistic effects which also go beyond the simple pencil drawing effect.

REFERENCES

1. Cabral, B., and Leedom, L. C. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM (1993), 263–270.
2. Castleman, K. R. Digital image processing, 1996. *Curve and Surface Fitting*, 501–507.
3. Kang, H., Lee, S., and Chui, C. K. Flow-based image abstraction. *Visualization and Computer Graphics, IEEE Transactions on 15*, 1 (2009), 62–76.
4. Kim, G., Kim, T., Lim, D.-H., and Yim, C. Sketch filter for feature extraction and rendering applications. *Electronics Letters 49*, 13 (2013), 805–806.
5. Kim, G., Woo, Y., and Yim, C. Color pencil filter for non-photorealistic rendering applications. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, IEEE (2014), 1–2.
6. Lee, H., Kwon, S., and Lee, S. Real-time pencil rendering. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ACM (2006), 37–45.
7. Lu, C., Xu, L., and Jia, J. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, Eurographics Association (2012), 65–73.
8. Mao, X., Nagasaka, Y., and Imamiya, A. Automatic generation of pencil drawing from 2d images using line integral convolution. In *CAD/Graphics*, vol. 9 (2001), 240–248.
9. Sousa, M. C., and Buchanan, J. W. Computer-generated graphite pencil rendering of 3d polygonal models. 195–208.
10. Sun, S., and Huang, D. Efficient region-based pencil drawing.
11. Way, D.-L., Yang, M.-K., Shih, Z.-C., and Lee, R.-R. A colored pencil non-photorealistic rendering for 2d images.
12. Webb, M., Praun, E., Finkelstein, A., and Hoppe, H. Fine tone control in hardware hatching. In *Proceedings*

of the 2nd international symposium on Non-photorealistic animation and rendering, ACM (2002), 53–ff.

13. Xie, D.-e., Zhao, Y., Xu, D., and Yang, X. Convolution filter based pencil drawing and its implementation on gpu. In *Advanced Parallel Processing Technologies*. Springer, 2007, 723–732.