11-18-2009

Adaptive Difference of Gaussian Algorithm for Coherent Line Drawing

Abadi Kurniawan
University of Missouri-St. Louis, aknn3@umsl.edu

ADAPTIVE DIFFERENCE OF GAUSSIAN ALGORITHM FOR COHERENT LINE DRAWING

by

ABADI KURNIAWAN

B.S. in Computer Science

A THESIS

Submitted to the Graduate School of the

UNIVERSITY OF MISSOURI- ST. LOUIS
In partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

November, 2009

ADAPTIVE DIFFERENCE OF GAUSSIAN ALGORITHM FOR COHERENT LINE DRAWING

by

Abadi Kurniawan

Submitted to the Department of Math and Computer Science
on November 03, 2009, in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE

Abstract

Non-photorealistic rendering is a method of imitating hand-drawn images using computer as the tool. One of hand-drawn techniques used by artists is line drawing. This goal of this thesis is to produce a technique to create line drawing images from real life picture. Based on a previous technique called Flow-based Difference of Gaussian (FDoG), we try to improve the output image so that it will create more believable pictures, as it if were hand-drawn by an artist.

FDoG has proven to be able to produce results with coherent and connected lines, but it fails to capture coarse details on isotropic areas in the image. Another technique in line drawing called Difference of Gaussian (DoG), which FDoG was based on, can produce better detail on isotropic areas. Combining these two techniques can create better results for both isotropic and anisotropic areas. We create an image segmentation technique using polarity to divide isotropic and anisotropic areas in the image. Using this segment, we then adaptively apply FDoG and DoG filters to each segment.

Thesis Supervisor: Henry Kang, Ph.D.
Title: Chairperson

Contents

List of Figures

Chapter 1

Introduction

Over the last 30 years, computer graphics researchers have tried to achieve photorealistic images using computer programs [17]. Ivan Sutherland was one of the pioneers in creating computer generated photorealistic images, working in the early 1960s. Photorealistic computer graphics is essentially an attempt to imitate real life pictures that would normally be taken by using another tool such as camera. Such attempt usually occurs in 3 dimensional (3D) computer graphics which, in most cases, tries to render the 3D object as realistically as possible. At the early stage of computer graphics research, these attempts has failed many times, especially on the attempts to imitate human appearance. Recent research, however, has started to show more believable lifelike human facial animation [11]. More recently, instead of producing results that look like real life objects, researchers have tried to produce more cartoon-looking or hand-drawn images; this technique is known as Non-Photorealistic Rendering.

Non-Photorealistic Rendering (NPR) is a process using a computer program to produce non-photorealistic renditions [17], such as sketch illustration, oil painting, watercolor-painting, etc. While using photorealistic images or even real photographs may deliver accurate information to the viewers, it also possesses some weaknesses, which can be surpassed by hand-drawn images. In some cases, hand-drawn images can actually delivers more information because they can allocate more focus on a certain aspect of a picture while ignoring unimportant ones. This way, the viewers will focus their attention on what the picture tries to convey. These kinds of images can be found in technical illustrations, e.g., in user manual of some electronic devices, or can also be found in medical illustrations.

Hand-drawn images can also be used to show the preliminary stage of a design in design phase period, for example, in architectural design [13], architects usually use pencil sketch drawings to show a draft of a design. The viewer can get the feeling of this preliminary stage better when see a hand-drawn pencil sketch image compared to photorealistic image.

Another benefit of using hand-drawn images is to give aesthetic feeling to an image. Images, such as paintings created by an artist, usually have aesthetic values that can stimulate certain emotions in humans. While the paintings themselves might not look as realistic as a photograph, they usually can stimulate senses and emotions in the people who see it.

The goal of NPR is not to replace human to create hand-drawing image, but to try to achieve the same effect of hand-drawn images in cases where actual hand-drawn image is very difficult or impossible to be created. The source of image generation on NPR can come from either 3 dimensional model or photographs taken by camera.

Since the first time NPR was introduced, there have been many researches work on this topic. Many of them have created techniques of imitating different styles of hand-drawing, e.g., sketch rendering [16], pen-and-ink illustration [18],[12], stipple drawing [14],[4], painterly effect [9],[7],[6], and line drawing [3], [15],[8]. In the next chapter, I discuss the background theories used in thesis, including Gaussian function, Difference of Gaussian technique (DoG), Flow-based Difference of Gaussian (FDoG). In chapter 3 and 4, I discuss about my attempt to improve the results of previous work by combining DoG and FDoG. In chapters 5 and 6, I discuss about results and conclusions of this experiment.

Chapter 2

Background

2.1 Line Drawing

Line drawing is a style of drawing that is commonly used to create illustration, such as technical illustrations, medical illustrations, caricature drawings, sketch drawings, etc. Line drawing uses straight or curved lines to form an image. Some techniques, such as stippling or hatching, can also be used to show different levels of shade. Line drawing has become one of the primary branches of NPR. Techniques have been improved in creating more convincing results as if it were actually drawn by an artist.

Line drawing in NPR can be divided into two categories based on the source of the image, which can be either a 3D object or a 2D object. Many works have been introduced on line drawing based on 3D object, e.g., by [3], [15]. Creating line drawing images from 3D models is relatively easier compared to 2D objects. In 3D models, the edge can be easily identified because all the information is stored in a straightforward manner, while in 2D objects, such as in photographs, information about edges are more difficult to extract and require extra steps and complicated methods to achieve a good result.

Most works in line drawing in NPR is based on edge detection or image segmentation techniques. The difference between edge detection and line drawing is that edge detection is more focused on the accuracy of edges extracted from pictures without considering the aesthetic aspect of how the lines (or edges) are presented. The goal of line drawing is to create a perceptually meaningful image that able to communicate the subject to the viewers. Line drawing in NPR tries to extract edges from pictures and forms lines, and it also tries to create artistic results, as if the image were created by artist. Some of the aspects that need to be considered in order to produce artistic results are: the structures of the lines, and the use of stylistic edges. There are many methods for edge detection, two examples of them are Canny's method [2], and Marr-Hildreth's method [10], which became the basis of Difference-of-Gaussian (DoG) filter and used by Gooch et al. in their facial illustration system to create some artistic line drawing, and by Winnemoller et al. to create a real time video abstraction.

2.2 Difference of Gaussian (DoG)

Gooch et al. in their facial illustration system [5] were able to create some artistic results. They were able generate a caricature of a human face based on photograph. Their method uses DoG filter, which was based on Marr-Hildreth's edge detection. DoG was also used by Winnemoller et al. in their video abstraction technique [19]. They use similar technique to Gooch et al., with addition of color, real-time performance, and temporal coherence.

The basic idea of DoG is to apply 2 Gaussian filters with different blur levels (σ). The second Gaussian filter uses a bigger blur radius to create blurrier image compared to the image produced by the first Gaussian filter. The area near the edge will have more intensity level difference than area far from edge when it is applied with Gaussian filter with different σ. The difference of intensity of each pixel of these two blurred images will be used by DoG to obtain edges, as shown in Figure 2-1.

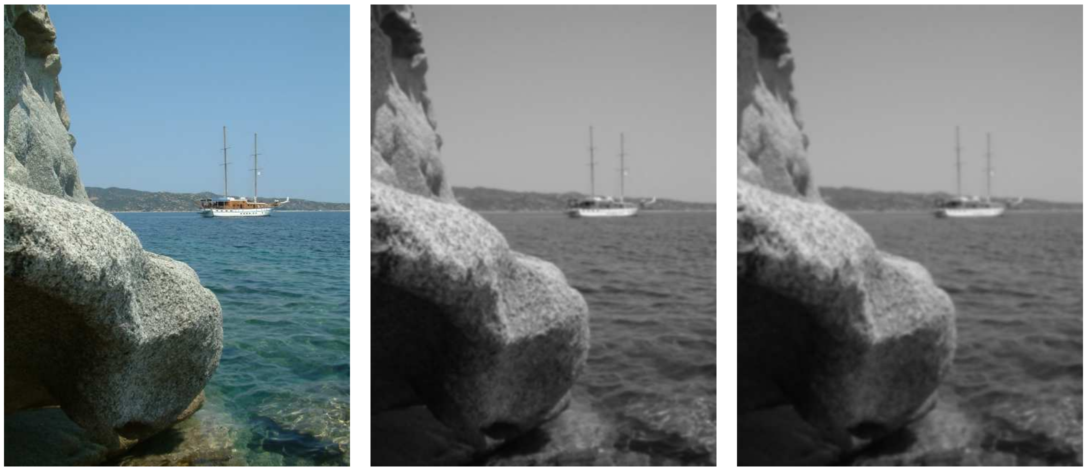
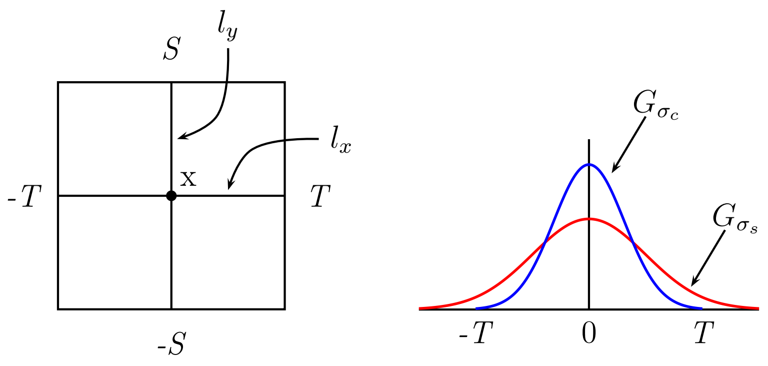

(a) Input image (b) Gaussian with $\sigma = 1.0$ (c) Gaussian with $\sigma = 1.6$

Figure 2-1: Sample of Gaussian filter

The Gaussian function used in DoG is formulated as the following 1-dimensional Gaussian function:

$$G(i,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{i^2}{2\sigma^2}}, \tag{2.1}$$

where σ is the blur level. $G(i,\sigma)$ is used as weight factor on how intensity of each pixel and its neighbor will influence intensity of the output, thus creating a blurred image.



(a) DoG Kernel (b) Gaussian components for DoG

Figure 2-2: Difference of Gaussian filtering

This weight factor with σ_c is first applied to each pixel's intensity along the l_x line started from $-T$ to T:

$$f(t,\sigma) = \int_{-T}^{T} G(t,\sigma) I(l_x(t))\, dt, \tag{2.2}$$

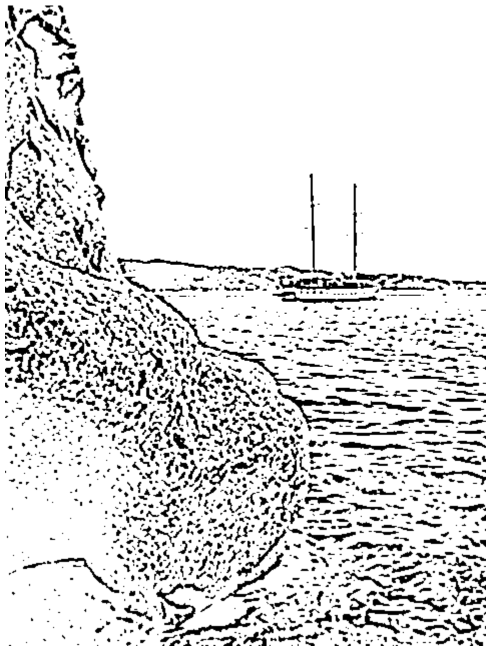
and then apply the same function to the value from $f(t,\sigma)$ along l_y line starting from -S to S, and repeat this step for every pixels on input image to produce the first blurred image.

$$F(\mathrm{x},\sigma) = \int_{-S}^{S} G(s,\sigma) f(l_y(s),\sigma)\, ds \tag{2.3}$$

The same procedure using different σ ($\sigma_s = 1.6\sigma_c$) is then applied to the same input image, producing the second blurred image. From these 2 output images, edges can then be identified by calculating differences of pixel intensity on these 2 images. Pixel with high intensity difference is much likely to be an edge (Figure 2-3).

$$H_d(\mathrm{x}) = F(\mathrm{x},\sigma_c) - \rho \cdot F(\mathrm{x},\sigma_s) \tag{2.4}$$

We use weight factor ρ to control how much the second blurred image contributes to the end results. Lower value means the less contribution on the end results, and vice versa.


(a) Rock and Boat

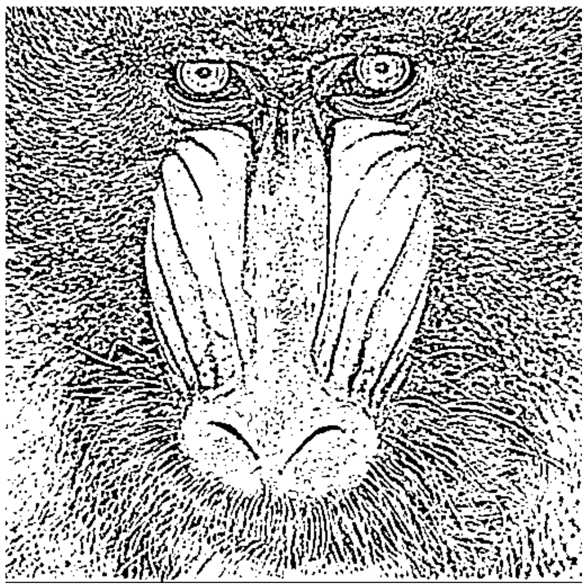

(b) Baboon

Figure 2-3: Difference-of-Gaussian

2.3 Flow-based Difference of Gaussian

Difference-of-Gaussian (DoG) method has the nature of isotropic because the kernel moves to every direction in order to calculate intensity differences. This behavior, when applied to images with anisotropic characteristic, will create a lot of discontinuity lines around edge because it fails to follow the path of the edge. So, instead of moving to every direction, Kang et al. suggested to use direction guide for the filter movement [8].

To create the direction guidance for DoG filter, Kang et al., use the information from edge flow, which is the direction in which exist the biggest contrast, and then run the DoG filter perpendicular to the edge flow direction [8]. By providing guide for the filter direction, Kang et al., were able to produce more coherent and connected line drawing.

2.3.1 Edge Tangent Flow

Kang et al., technique in creating line drawing required two steps. First step is to generate edge tangent flow that will provide direction guide for DoG filter. To create edge tangent flow from input image $I(\mathrm{x})$, where $\mathrm{x} = (x, y)$ denotes a pixel at coordinate (x, y), Kang et al., use edge tangent $\mathrm{t}(\mathrm{x})$ as a vector perpendicular to the image gradient $g(\mathrm{x}) = \nabla I(\mathrm{x})$, and called it Edge Tangent Flow (ETF).

They presented a technique to construct ETF by using kernel-based nonlinear vector smoothing of vector field. The ETF construction filter is as follows:

$$\mathrm{t}^{new}(\mathrm{x}) = \frac{1}{k} \sum_{\mathrm{x}' \in \Omega(\mathrm{x})} \phi(\mathrm{x}, \mathrm{x}')\mathrm{t}^{cur}(\mathrm{x}')w_s(\mathrm{x}, \mathrm{x}')w_m(\mathrm{x}, \mathrm{x}')w_d(\mathrm{x}, \mathrm{x}'), \tag{2.5}$$

where k is the vector normalizing term and $\Omega(\mathrm{x})$ is the neighbor of x. ETF function uses 3 weight functions. The first one is w_s, a *spatial weight function*, which uses radially-symmetrical box filter with radius of r, where r is the radius of kernel Ω. The *spatial weight function* w_s is defined as follows:

$$w_s(\mathrm{x}, \mathrm{x}') = \begin{cases} 1 & \text{if } \|\mathrm{x} - \mathrm{x}'\| < r \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The second weight function is w_m, a *magnitude weight function*, which is defined as follows:

$$w_m(\mathrm{x}, \mathrm{x}') = \frac{1}{2}(1 + \tanh[\eta \cdot (\hat{g}(\mathrm{x}') - \hat{g}(\mathrm{x}))]), \tag{2.7}$$

where $\hat{g}(\mathrm{x})$ and $\hat{g}(\mathrm{x}')$ denote normalized gradient magnitude at x and x′ respectively. Variable η is the weight variable that determine how the neighboring pixel x′ with gradient value higher than gradient value of x will influence the edge tangent t(x).

The third weight function is the direction weight function w_d, which is defined as follows:

$$w_d(\mathrm{x}, \mathrm{x}') = |\mathrm{t}^{cur}(\mathrm{x}) \cdot \mathrm{t}^{cur}(\mathrm{x}')|, \tag{2.8}$$

where $\mathrm{t}^{cur}(\mathrm{x})$ and $\mathrm{t}^{cur}(\mathrm{x})$ denote the current normalized tangent vector at x and x′ respectively. This function uses a dot product between tangent vectors at a particular location and its neighbors. The cross product produces a high weight value for vectors that are closely aligned (where angle between two vectors (θ) is close to 0° or 180°) and produces a low weight value for vectors that are perpendicular to each other (where angle between two vector (θ) is close to 90°).

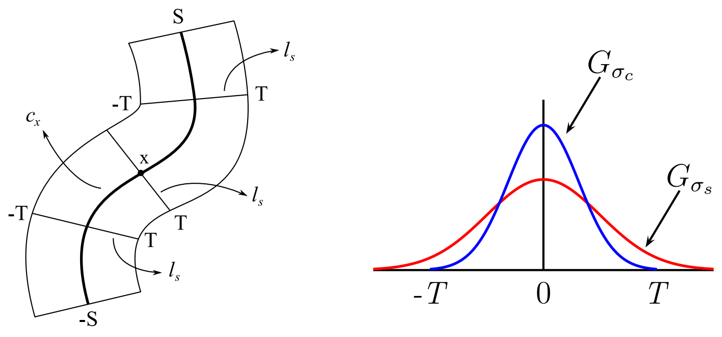
The last function is used to reverse the direction of $\mathrm{t}^{cur}(\mathrm{x}')$ when angle between vectors (θ) is bigger than 90° (dot product between two vectors is ¡ 0):

$$\phi(\mathrm{x}, \mathrm{x}') = \begin{cases} 1 & \text{if } \mathrm{t}^{cur}(\mathrm{x}) \cdot \mathrm{t}^{cur}(\mathrm{x}') > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.9}$$

As shown in the function, $\mathrm{t}^{new}(\mathrm{x})$ is achieved from applying all functions to $t^{cur}(\mathrm{x})$, and initially, $\mathrm{t}^{\circ}(\mathrm{x})$ is achieved from normalized vector perpendicular to gradient $g(\mathrm{x})$ in gradient map of the input image I. ETF is constructed by iteratively applies t(x) function several times (in their experiment, Kang et al. applied the function 2 or 3 times).

2.3.2 Flow-based DoG

After constructing ETF, the next step will be applying anisotropic DoG with edge flow information from ETF to guide its direction. Kang et al. built kernel with shape based on local flow from ETF, and applied DoG on this kernel. An example of kernel is shown in Figure 2-4. Line c_x is a line that follows direction of flow from ETF, and line l_s is a line tangent to line c_x.



(a) FDoG Kernel (b) Gaussian components for DoG

Figure 2-4: Flow-based DoG Kernel

First step of FDoG is to apply 1 dimensional filter $f(t)$ along the line l_s:

$$F(s) = \int_{-T}^{T} I(l_s(t)) f(t)\, dt, \tag{2.10}$$

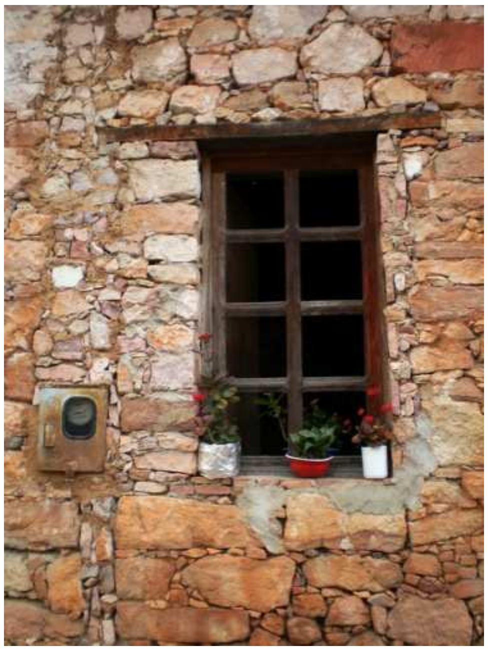
where $l_s(t)$ denotes a pixel on the line l_s at parameter t, and $I(l_s(t))$ is intensity level of the pixel. Function $f(t)$ is the same DoG function from previous chapter that uses 2 Gaussian filters with different σ and then calculate the differences of 2 output values.

$$f(t) = G_{\sigma_c}(t) - \rho \cdot G_{\sigma_s}(t) \tag{2.11}$$

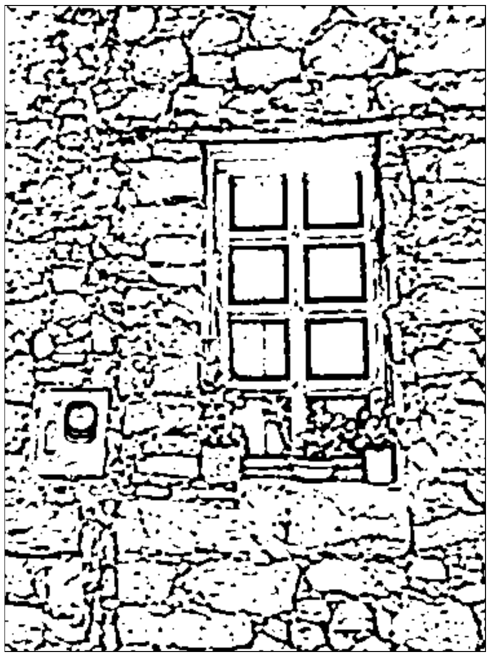
The ρ value controls how much the second blurred image contributes to the end results, and in their experiment, Kang et al. used value of 0.99. After applying filter $f(t)$ to each pixel, filter $F(t)$ is applied along the line c_x from $-S$ to S.

$$H_f(\mathrm{x}) = \int_{-S}^{S} G_{\sigma_m}(s) F(s)\, ds, \tag{2.12}$$

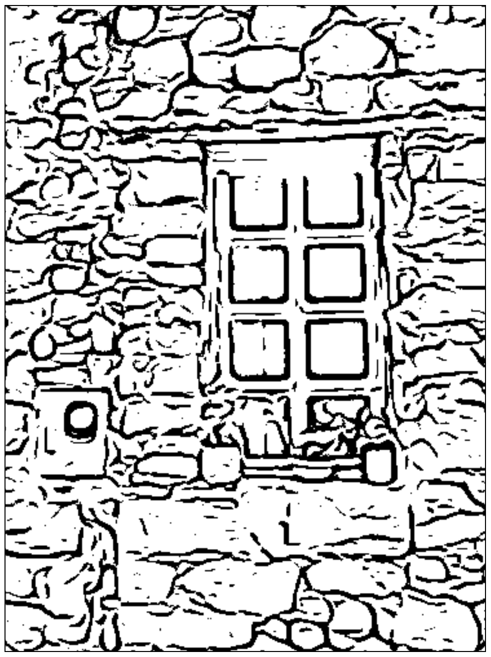
where G_{σ_m} is another Gaussian function that will become weight factor for each response according to s. As shown in Figure 2-5, image from FDoG has continuous lines and less noise compared to DoG.



(a) Input image



(b) DoG



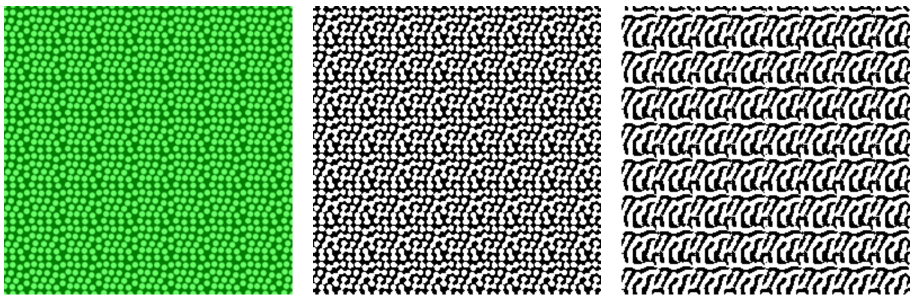
(c) FDoG

Figure 2-5: Flow-based Difference-of-Gaussian

Chapter 3

Hybrid-DoG

Images created by FDoG contain more coherent and connected lines compare to images created by DoG. The directional kernel used in FDoG has been proven to be able to create well connected lines on anisotropic image, thus creates good line drawing image. One major drawback of FDoG is it cannot capture all the details on image with isotropic characteristic. Example on Figure 3-1 shows that FDoG fails to capture the details of dotted textures, and creates lines in unexpected area. While DoG can easily overcome this problem, it still cannot outmatch FDoG in cases with anisotropic characteristic images. These two different behaviors of FDoG and DoG on handling isotropic and anisotropic images bring us into the idea of combining the two methods in order to achieve optimal results in both cases.



(a) Input Image (b) DoG (c) FDoG

Figure 3-1: Weakness of FDoG

3.1 Combining FDoG with DoG

Our initial and naive method to combine DoG and FDoG is by combining the results of FDoG and DoG, and we name it Hybrid-DoG:

$$H_h(\mathrm{x}) = \lambda H_f(\mathrm{x}) + (1 - \lambda)H_d(\mathrm{x}) \tag{3.1}$$

with λ value ranges between 0 and 1. Value of λ will determine the closeness of the output to FDoG or DoG. When the value of λ approaches 0, it will produce output that similar to DoG, and will produce output similar to FDoG when it approaches 1.

With this variable λ, we can control the behavior of the filter according to the character of input images. Use value of λ close to 0 for input images with majority of isotropic images, and use value of λ close to 1 for anisotropic images. For images with both isotropic and anisotropic characteristic, based on our experiment, values of 0.5 for λ will produce good results for most images.

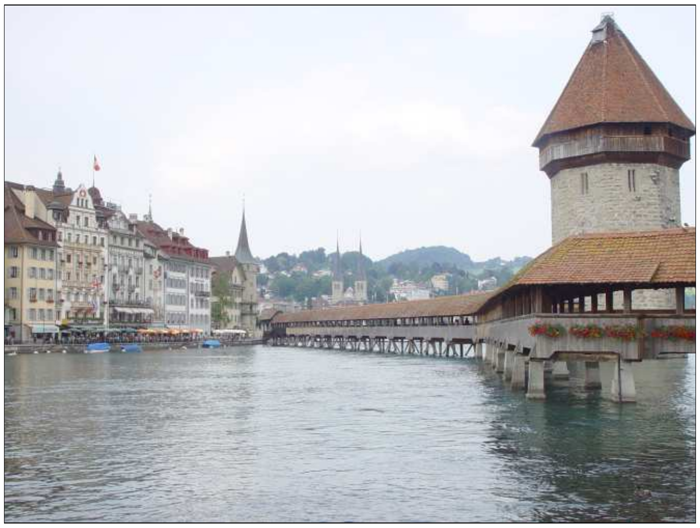
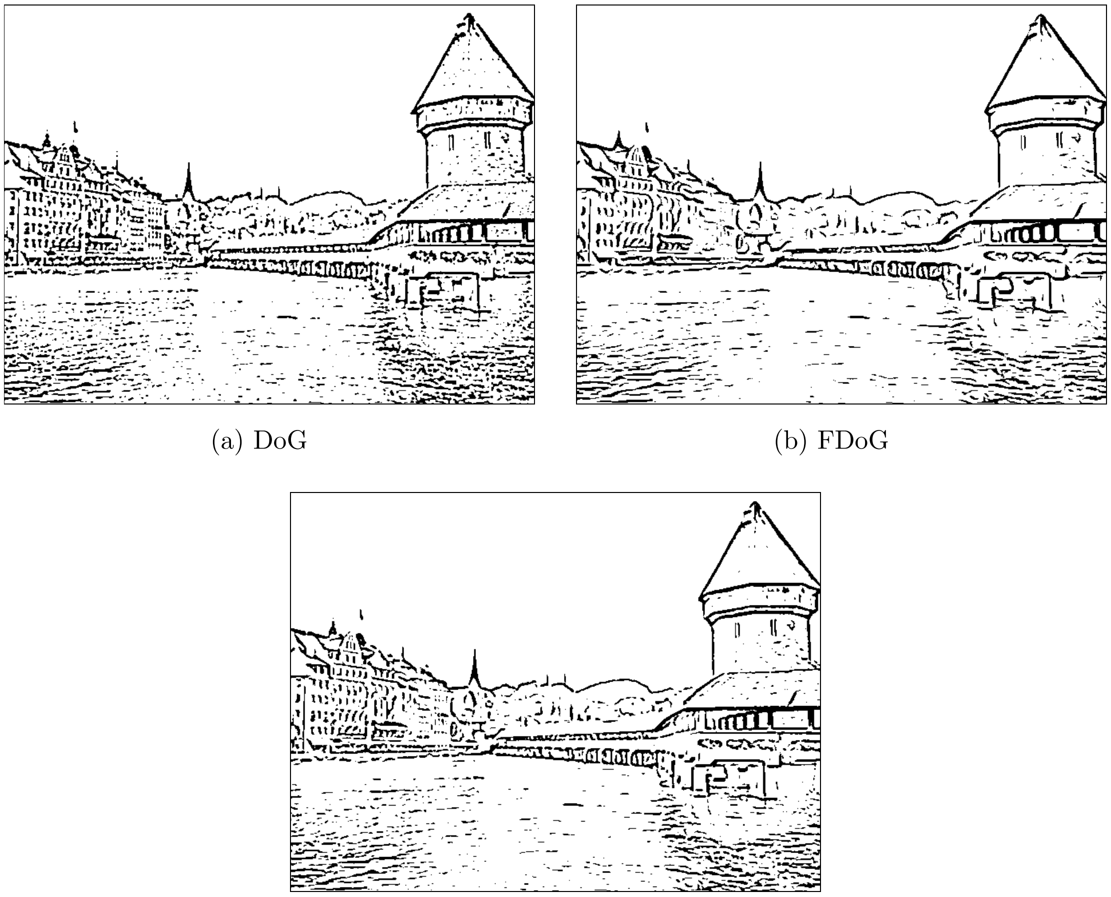

(a) Input image

Figure 3-2: Copenhagen


(a) DoG (b) FDoG

(c) Hybrid-DoG

Figure 3-3: Hybrid-DoG compared to DoG and FDoG

Figure 3-2 is a picture of boats with houses in the background. Comparison between DoG, FDoG and our Hybrid-DoG method can be found in Figure 3-3, where we can see that both DoG and FDoG have strength and weakness for this image. In DoG, we can see all detail on the boats and windows, but in the water area, we can see a lot of disconnected line. In FDoG, although we can see more connected line in the water and the houses, all the details on the boat and the windows have been distorted by the generated lines. In our Hybrid-DoG method, we certainly can see some improvement from both DoG and FDoG, where more lines appears to be connected, and more details are preserved.

Chapter 4

Adaptive-DoG Technique

Although hybrid method has shown improvement from DoG and FDoG, this method still has weakness. Hybrid-DoG method simply cannot produce output as good as DoG on isotropic image, nor it can produce output as good as FDoG on anisotropic image since it basically combine both FDoG and DoG regardless of the characteristic of the input image.

We proposed a more intelligent method to combine DoG and FDoG by selectively apply the right function for different area depending on its characteristic. Area with isotropic textures will use DoG to retain the details, while area with anisotropic detail will use FDoG to create more coherent lines and reduce noise. To be able to adaptively apply the right filter, we first need to distinguish between isotropic and anisotropic segment in an image. Belongie et al., [1] in their paper used polarity to segment the image based on texture type. In this experiment, we simplify the technique used by Belongie et al.

4.1 Image Segmentation

Polarity is calculated by comparing the gradient direction of a pixel to its neighbor's gradients direction. Area with similar gradient vector direction is considered as high polarity area, while area with non-uniform gradient vector direction is considered as low polarity area. Non-uniform gradient vector direction can be found in isotropic area where the gradients in this area are scattered. Anisotropic area will have more uniform and similar direction, although it may also have gradients with direction opposite to each other as we can see on area with stripe pattern. Using this behavior, we can distinguish between isotropic and anisotropic area by determining the polarity of each pixel. One exception that needs to be considered is isotropic area with low gradient magnitude should be considered as isotropic segment, since this area should be applied with FDoG filter to minimize noise.

To calculate polarity, we use a function S which will be applied to pixel x:

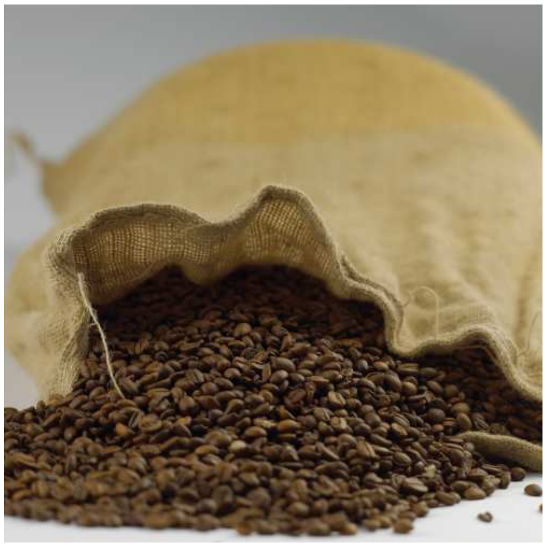
$$S(\mathrm{x}) = \frac{\sum\limits_{\mathrm{x}' \in \Omega(\mathrm{x})} \mid g(\mathrm{x}) \cdot g(\mathrm{x})' \mid}{\mid \Omega(\mathrm{x}) \mid}, \tag{4.1}$$

where $\Omega(\mathrm{x})$ is the neighborhood of pixel x and pixel x$'$ is element of $\Omega(\mathrm{c})$. The sum of dot product is then divided by $\mid \Omega(\mathrm{x}) \mid$, the total number of pixels in the neighborhood, to get the average.

The next step is segmenting the image using the polarity obtained from previous step:

$$P(\mathrm{x}) = \begin{cases} 0 & \text{if } S(\mathrm{x}) \geq \alpha \text{ or } |g(\mathrm{x})| < \beta \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

We use a threshold α as a cut point of polarity level, and in our experiment, we use $\alpha = 0.875$. Area with polarity over the threshold will be considered as anisotropic segment (labeled with 1). For isotropic segment (labeled with 0), we consider isotropic area with low gradient magnitude as anisotropic segment (labeled with 1). Variable β is the threshold for gradient magnitude at pixel x and we use $\beta = 0.1$ in this experiment. Figure 4-1 shows example of polarity segment created by our segmentation method.


(a) Input image

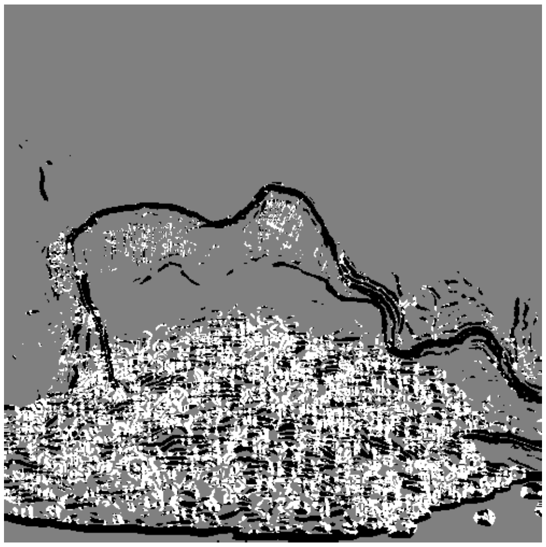

(b) Polarity-based image segment; white indicates isotropic area (label 0), black indicates anisotropic area (label 1), and gray indicates area with low gradient magnitude (label 1)

Figure 4-1: Image Segmentation

4.2 Adaptive-DoG

Once we obtain the image segment, the next step to adaptively apply hybrid DoG will be trivial. Provided the characteristic of each area, we are now able to determine the proper filter for each different area by using labels from image segmentation step.

$$H_a(\mathrm{x}) = \begin{cases} H_d(\mathrm{x}) & \text{if } P(\mathrm{x}) = 0 \\ H_f(\mathrm{x}) & \text{otherwise} \end{cases} \tag{4.3}$$

Image segment labeled with 0 (isotropic segment) will be applied with DoG filter H_d, while image segment labeled with 1 (anisotropic segment) will be applied with FDoG filter H_f.

Chapter 5

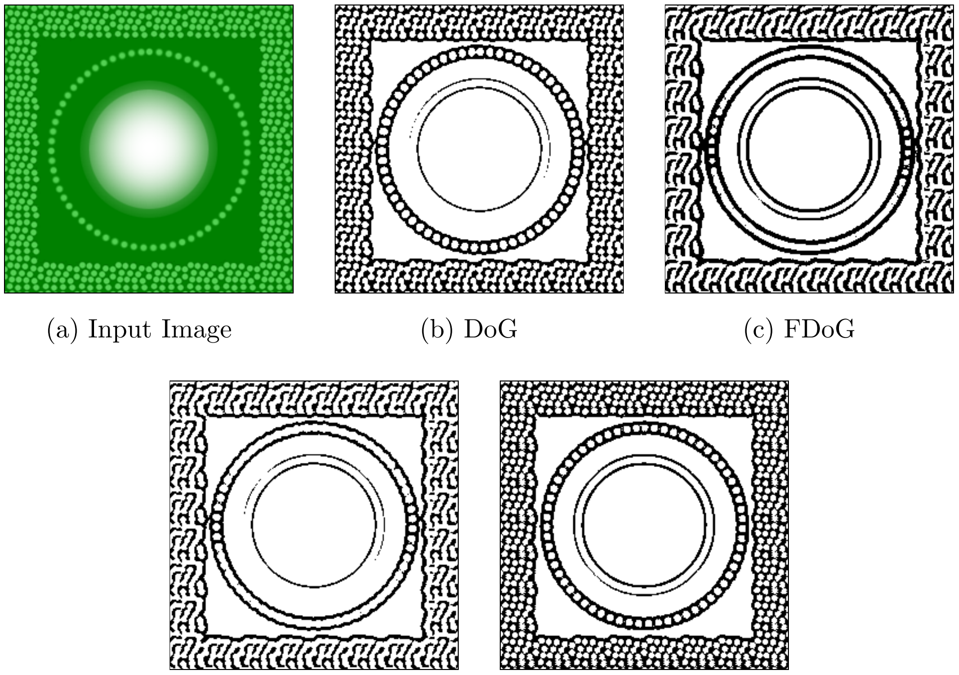
Results

In this thesis, we use 40 images as our test set, and we display sample of the results that represent different type of images. Our test set contains pictures with different type of object, e.g., nature photo, scenery, aerial photos, human face, and animals. We choose pictures that contains both anisotropic and isotropic characteristic to evaluate how our Adaptive-DoG method perform in cases where both DoG and FDoG perform poorly. Most of the pictures were downloaded from a free stock photography website called Stock.xchng (http://www.sxc.hu), and the rest are widely used pictures in the area of edge detection or NPR.

Evaluation method on Non-Photorealistic Rendering is inheritably subjective due to the lack of ground truth of the image. Each viewer may have different ground truth of what good image is, which will make objective evaluation difficult to be conducted. All evaluation in this thesis is limited to subjective evaluation.

We compare the results of Adaptive-DoG with the results of Hybrid-DoG, DoG, and FDoG to show how our method manages to outperform them in most cases. We also compare thin line version of our Adaptive-DoG method with Canny edge detection method to see how it perform as an edge detector.

To keep the comparison fair, we use the same parameters for all methods. Parameters used for DoG are as follows: $\sigma_s = 1.6\sigma_c$, and $\rho = 0.99$. For FDoG, we use same parameters from DoG for σ_c, σ_s, and ρ, with additional parameters as follows: ETF kernel size = 4 pixels, ETF iteration = 2. All parameters are also used in Hybrid-DoG and Adaptive-DoG since both methods use functions from DoG and FDoG.

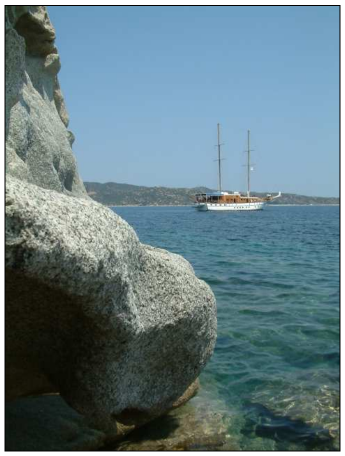


(a) Input Image (b) DoG (c) FDoG

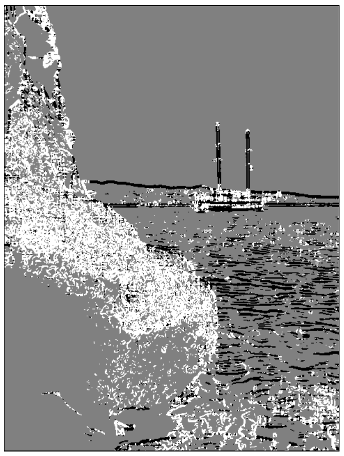
(d) Hybrid-DoG (e) Adaptive-DoG

Figure 5-1: Synthetic Image

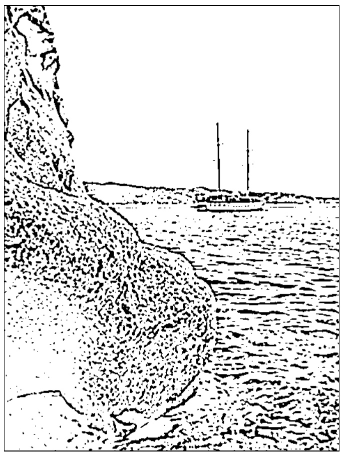
Using synthetic image on Figure 5-1, we compare the performance of Adaptive-DoG with DoG and FDoG in handling image with small dot pattern. Our adaptive technique was able to produce similar results with DoG and was able to maintain the dot pattern in the image. Our method even produced more definite line of the pattern compared to the line produced by DoG. We can see also that both FDoG and Hybrid-DoG produce connected lines on the dotted circle and lost almost all of the dot pattern that construct the middle circle, while both DoG and Adaptive-DoG produce more accurate representation of the circle. Our method also produced better connected circle of the circular gradient in the center of the picture.



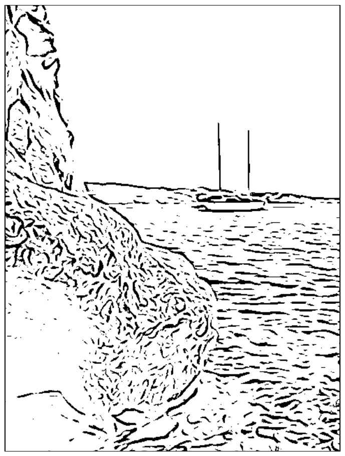
(a) Input Image



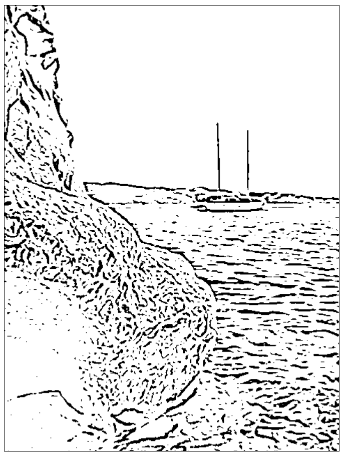
(b) Polarity Segment



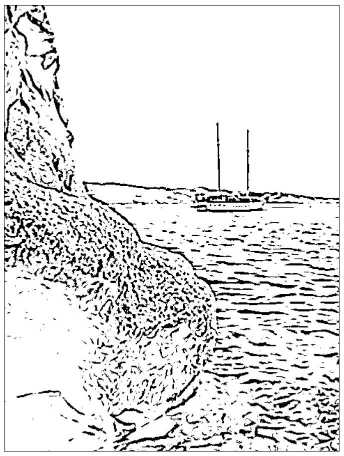
(c) DoG



(d) FDoG



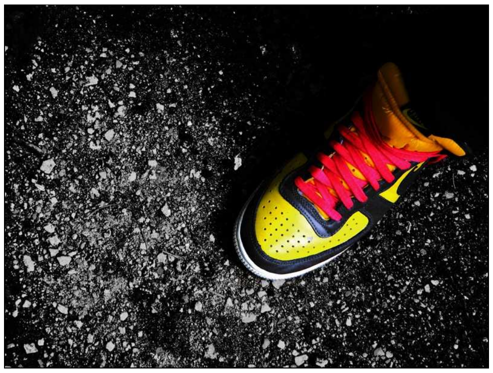
(e) Hybrid-DoG

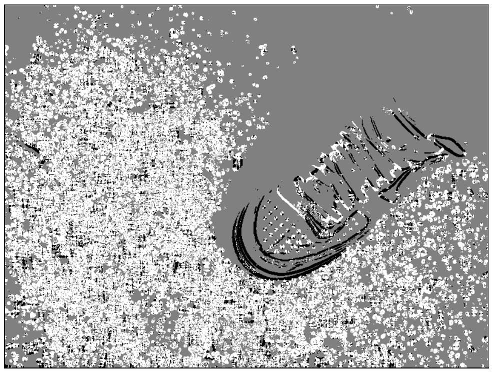


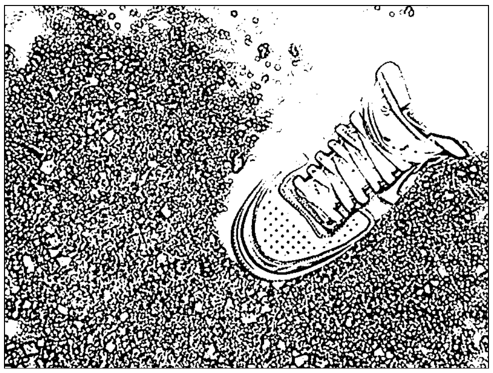
(f) Adaptive-DoG

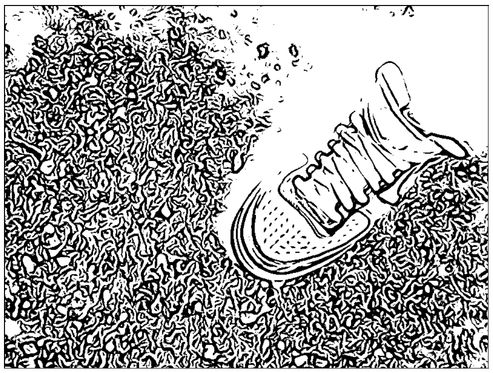
Figure 5-2: Boat and Rock

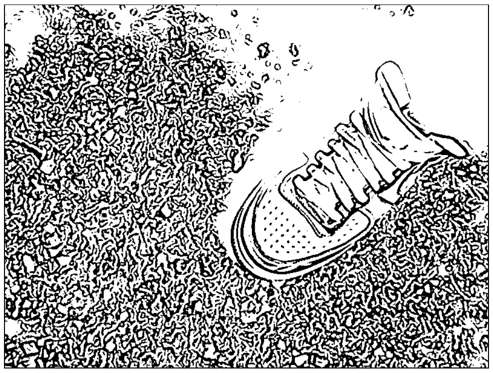
Figure 5-2 is a picture of rock and a boat on the sea. The rock on the left-side of picture contain coarse texture, which is detected as isotropic segment by the polarity-based segmentation, while the boat, the landscape, and some of the wave on the sea are detected as anisotropic segment. In this picture, FDoG failed to get the correct texture shape for the rock. Our hybrid method, although able to produced better texture of the rock, still shows some line shape which similar to FDoG. Our Adaptive-DoG was able to produce better looking texture of the rock, although small amount of lines still appear on area near the shadow. Even-tough Adaptive-DoG failed to exceed DoG on the rock texture, the wave on Adaptive-DoG still looks better and cleaner than DoG.


(a) Input Image


(b) Polarity


(c) DoG


(d) FDoG


(e) Hybrid-DoG

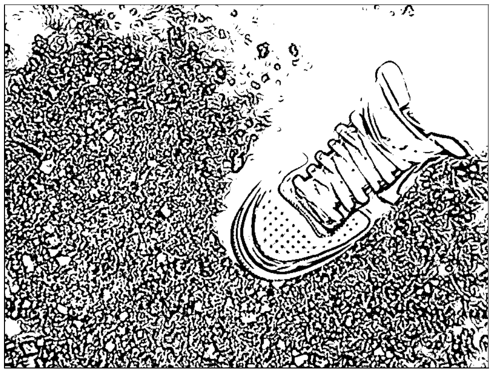
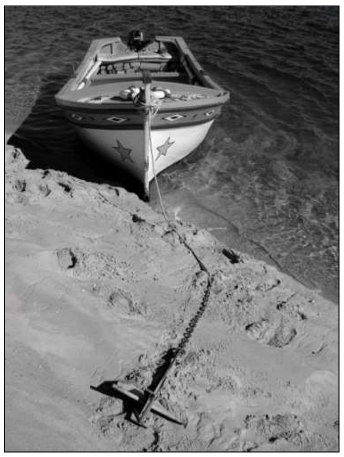
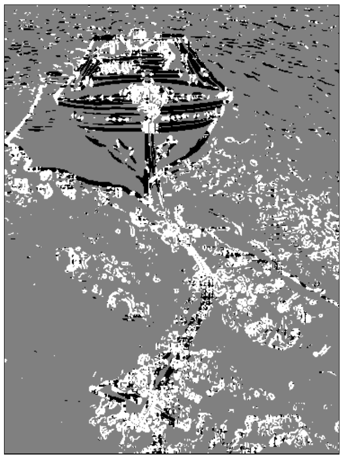

(f) Adaptive-DoG

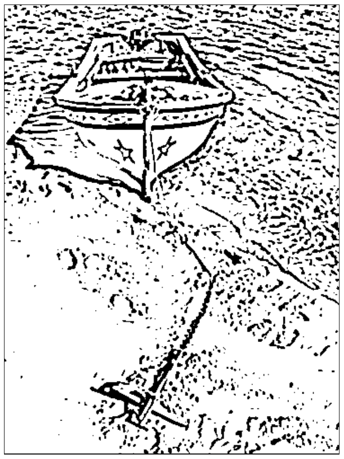
Figure 5-3: Shoe on Gravel

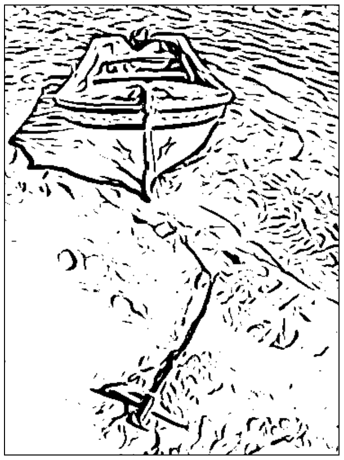
Figure 5-3 is a picture of shoe on gravel. This picture contains both anisotropic area, which is the shoe, and isotropic area, which is the gravel in the background. The results show that DoG is managed to accurately capture the gravel texture, while

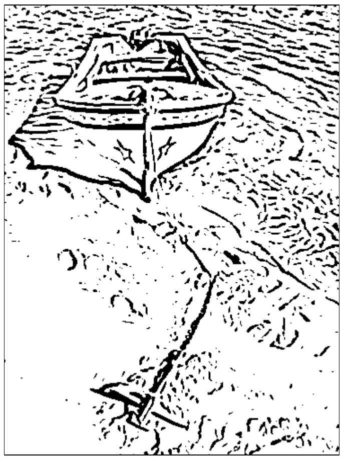
FDoG completely failed to do it. In Hybrid-DoG, although we see some improvement from FDoG, it is still not as good as the results of DoG. Our method successfully produced results similar with DoG, because our polarity segment managed to recognize the background gravel as isotropic segment. The shoe in Adaptive-DoG looks cleaner and the lines are more connected compared to DoG.


(a) Input Image


(b) Polarity Segment


(c) DoG


(d) FDoG


(e) Hybrid-DoG

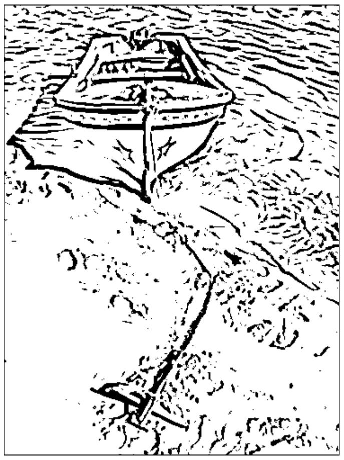
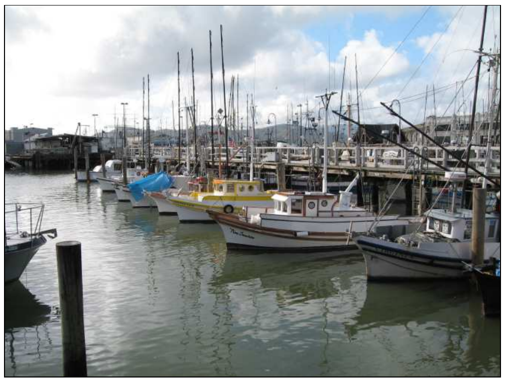
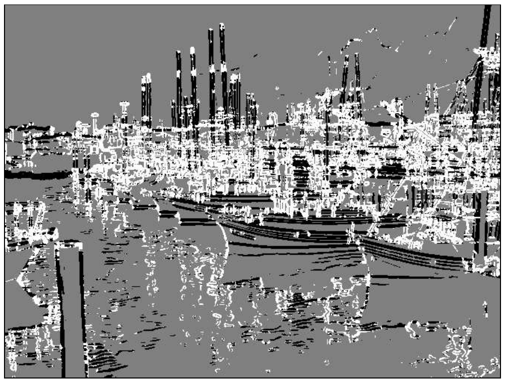
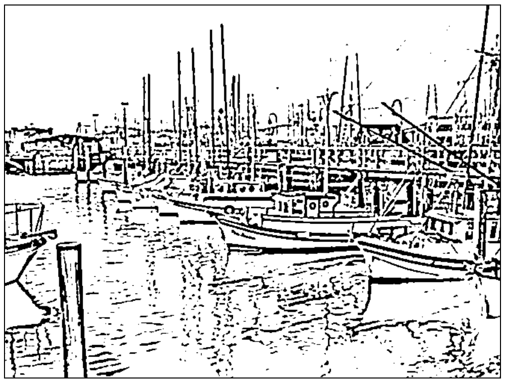

(f) Adaptive-DoG

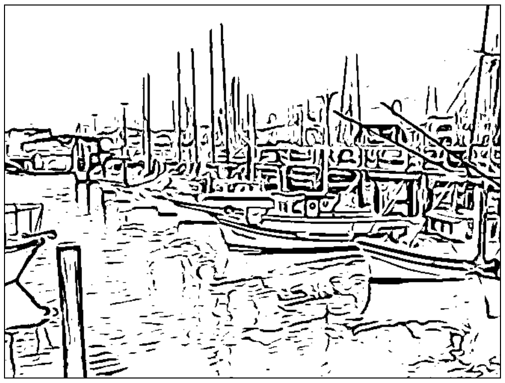
Figure 5-4: A Boat in Sand

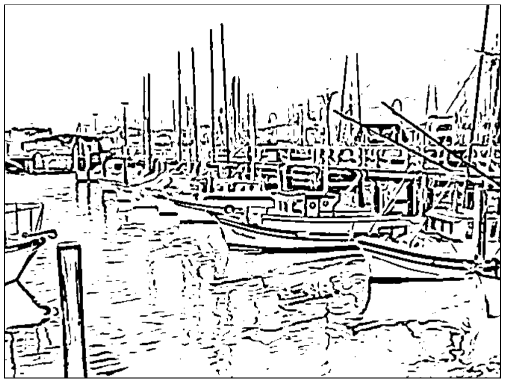
Figure 5-4 is similar to Figure 5-2. More coarse texture can be found on the beach sand, especially around the anchor. The results of hybrid method, similar to Figure 5-2 shows some improvement from both DoG and FDoG; however, Adaptive-DoG was able to produce even better results. The sand in Adaptive-DoG appears more similar to the actual sand than it does on FDoG. The boat and the wave look better compared to DoG.


(a) Input Image


(b) Polarity Segment


(c) DoG


(d) FDoG


(e) Hybrid-DoG

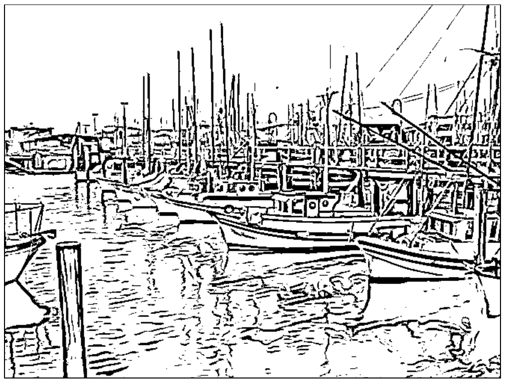
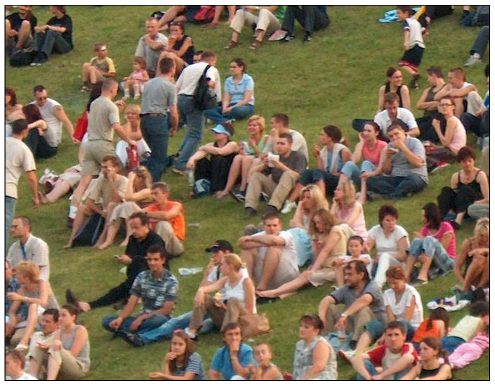
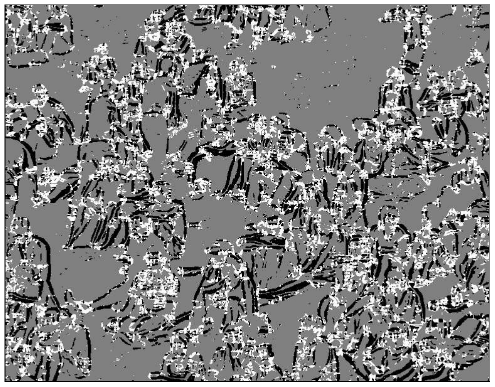
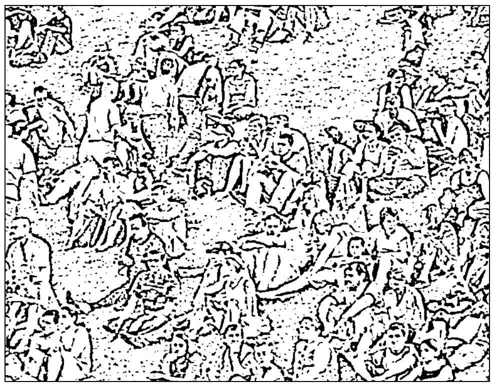

(f) Adaptive-DoG

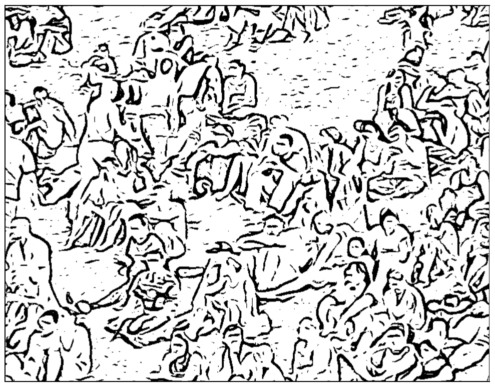
Figure 5-5: Boats at the Bay

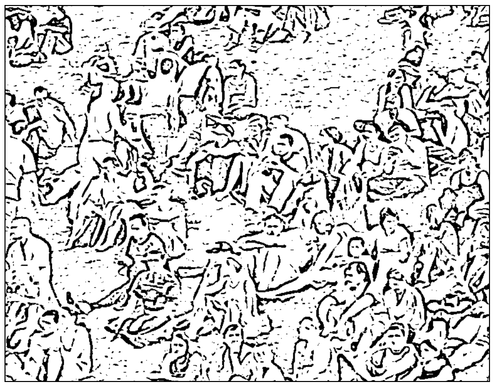
Figure 5-5 is a picture of sail boats anchored in a bay. In this picture, we can clearly see that Adaptive-DoG have better results than DoG and FDoG. They both have better connected lines than DoG. They both also have details in the bay area. We see slight improvement between Adaptive-DoG and Hybrid-DoG in the bay area where the image from Adaptive-DoG looks less distorted than Hybrid-DoG.


(a) Input Image


(b) Polarity Segment


(c) DoG


(d) FDoG


(e) Hybrid-DoG

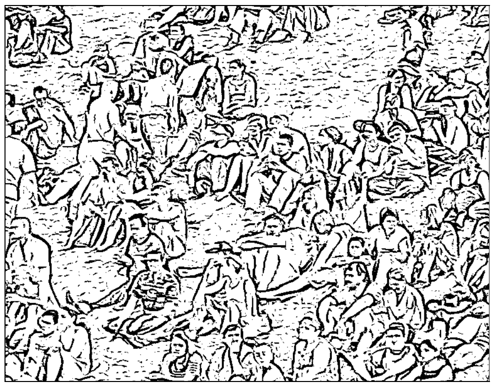

(f) Adaptive-DoG

Figure 5-6: Crowd

Figure 5-6 is a picture of crowd. In this picture, we can see that DoG was not able to produce good results; most of the line appears disconnected, and the image look noisy. FDoG produced well connected line, but failed to form a clear image of the people. Hybrid-DoG shows some improvement from both methods; however, Adaptive-DoG produced better results with well connected line, clear and detail image of the people.

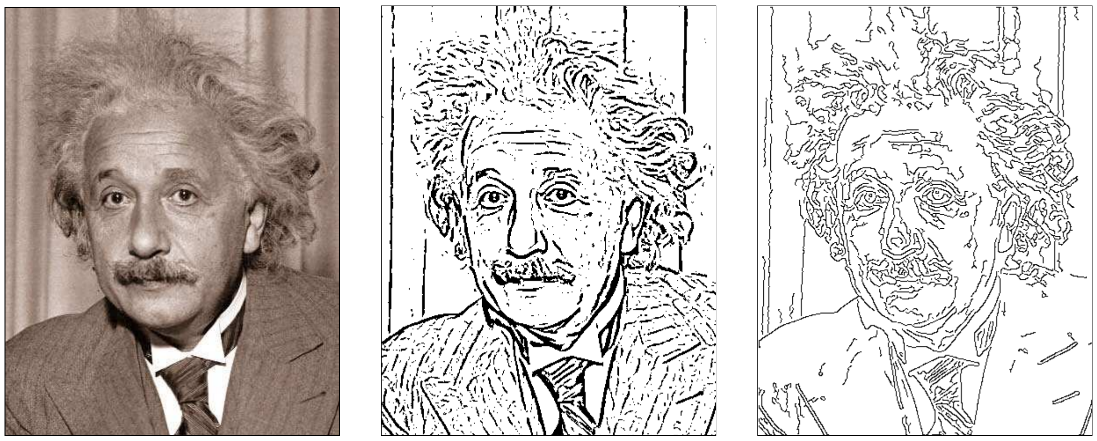
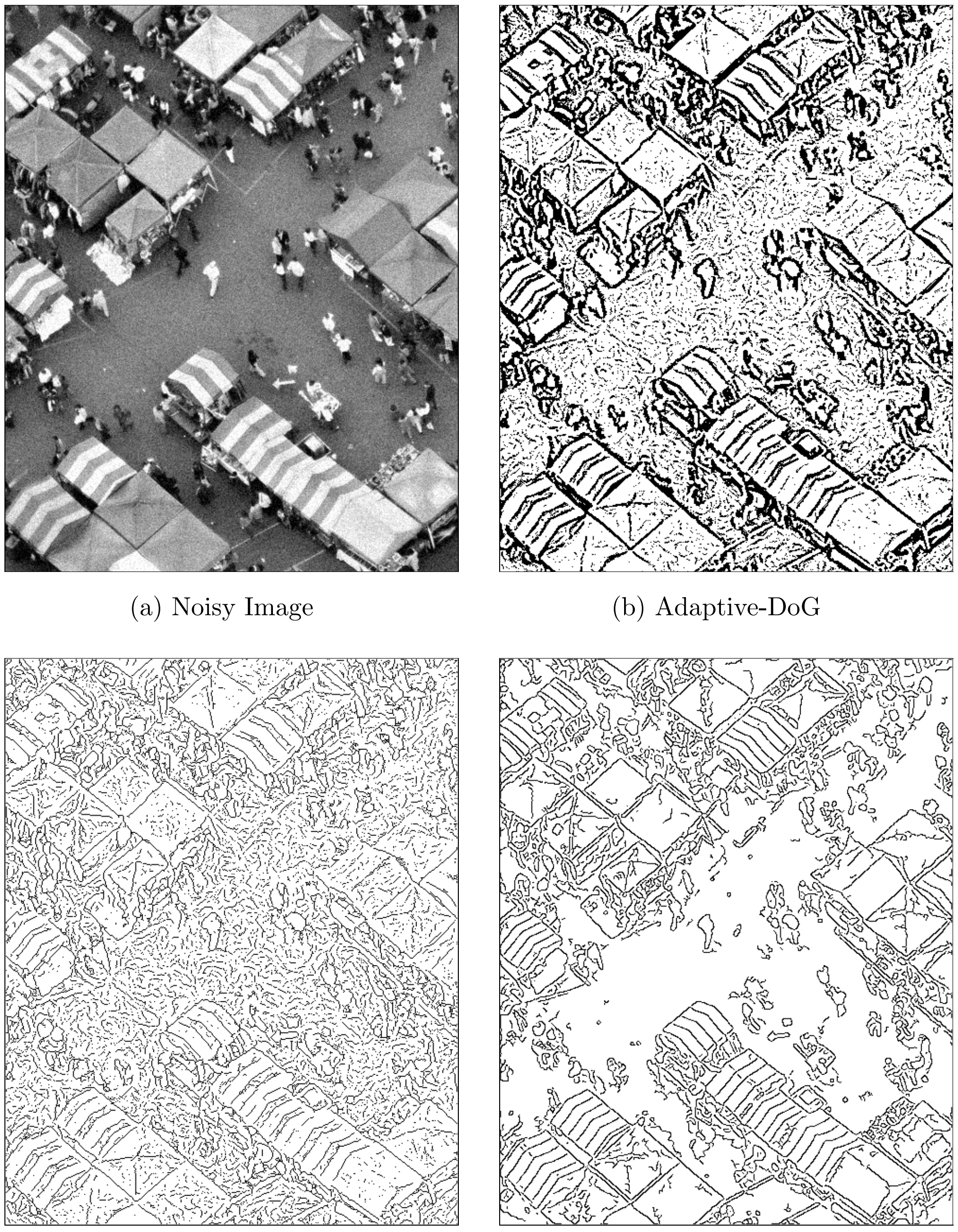

(a) Input Image (b) Adaptive-DoG (c) Canny Edge Detection

Figure 5-7: Comparison with Canny Edge Detection

Although the goal of our method is not to produce a good edge detection technique, it is interesting to know how our method performs as one. To evaluate the performance, we compare our method with a well known edge detector: Canny edge detector [2]. We use picture of human face as our first image. Canny's method is known to perform poorly in detecting edge of human face, mainly because the goal of Canny's method is to get a precise edge location. It does not concern about which line is more important than the other line; whereas the goal of our method is to communicate to the viewers about the subject in the picture. It is important to show which lines are more important and which lines are less important.

We can see in Figure 5-7, our method outperformed Canny's in detecting human face. We can see thicker line on the outline of the face, while the lines on the skin appear thinner. In Canny's, we can see a lot of unnecessary lines on the skin, and double lines around the area of eyes, nose, and mouth.

On our next image, we add noise to the original image to see how our method performs against noise. We add Gaussian white noise with standard deviation (σ) of 0.4 and intensity of 0.3. We also make another comparison by modifying our method using thin lines to make the output similar with output of Canny edge detector.


(a) Noisy Image (b) Adaptive-DoG

(c) Adaptive-DoG Thin Line (d) Canny Edge Detection

Figure 5-8: Comparison with Canny Edge Detection for noisy image

The results on Figure 5-8 shows that Canny edge detector is more robust in detecting edge in noisy image. Our method, especially the one with thin line, produced a lot of noises in the image, whereas canny managed to minimize the noise. The area between tents in canny is much cleaner than it is in our method. The thick line method also produced noises in the image, but we can see distinct thickness of the lines, where the noises use thinner line.

The current implementation of our Adaptive-DoG filter is not optimized for performance yet. Currently, it runs significantly slower than DoG and FDoG. Our implementation of polarity calculation required $1344ms$ to perform on image with $750 * 500$ dimension with size of neighborhood at $20 * 20$ pixels. The adaptive-DoG filter itself ran for $3390ms$ on the same image. The total run time for this technique is $7797ms$. This number is significantly higher than DoG, which only required $3859ms$, and also FDoG, which only required $4578ms$. The main problem from our implementation is that we still apply both DoG and FDoG filters, in which both output will be selectively chosen based on image segment. Further improvement of this implementation is required to make the program run faster and more efficient.

Chapter 6

Conclusion

Based on the results of this experiment, our method is able satisfy the goal of line drawing, which is creating a perceptually meaningful images. Our Adaptive-DoG technique is able to produce better results compared to DoG and FDoG. The lines appear to be more connected than lines in DoG images, and the details of coarse texture is better preserved than in FDoG images. Our method is also able to capture more detail of the picture, along with generating well connected lines on the edges.

Although the output is relatively better in some cases, the results of Adaptive-DoG may never exceed the results of neither DoG nor FDoG in some specific cases. For input image that only contains lines without any coarse texture, the results of Adaptive-DoG will, at best, be the same with FDoG, while for input image with coarse texture only, the results will never exceed DoG.

Although our method is not intended to be an edge detector, our experiment shows that it is able to perform edge detection and produce good results. It is able to outperform canny on some cases where precision of the edge detection is not very important as long as the results is able to communicate the subject to the viewer, e.g. image of human face. In cases where precision of the edge detection is important, canny is better than our method in showing edges and suppressing noise.

Another note from experiment that we need to mention is the performance in term of running time. Our experiment shows that the performance of the filter is relatively slower than both DoG and FDoG. Since the operation of Adaptive-DoG is local and operates on pixel by pixel, each calculation can be executed independently. This condition open the possibility for parallel programming using Graphics Processing Unit (GPU) in which case, will accelerated the running time drastically.

Possible future improvement for Adaptive-DoG is by iterating the filters to produce more solid shape for both lines and textures. The same polarity map may be use on the next iteration, but generating new polarity map might also produce better results.

Another possible future improvement is to integrate Hybrid-DoG technique on Adaptive-DoG, where DoG and FDoG is proportionally combined with a weight ratio determined by the polarity level of the image. Area with high level polarity will have weight ratio closer to DoG than FDoG, while area with low level of polarity will have weight ratio closer to FDoG.

Bibliography

[1] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using em and its application to content-based image retrieval. *Computer Vision, 1998. Sixth International Conference on*, pages 675–682, Jan 1998.

[2] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov. 1986.

[3] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[4] Oliver Deussen, Stefan Hiller, Cornelius Van Overveld, and Thomas Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19:40–51, 2000.

[5] Bruce Gooch, Erik Reinhard, and Amy Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.*, 23(1):27–44, 2004.

[6] James Hays and Irfan Essa. Image and video based painterly animation. *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120, 2004.

[7] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Com-*

puter graphics and interactive techniques, pages 453–460, New York, NY, USA, 1998. ACM.

[8] Henry Kang, Seungyong Lee, and Charles K. Chui. Coherent line drawing. *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 43–50, 2007.

[9] Peter Litwinowicz. Processing images and video for an impressionist effect. *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414, 1997.

[10] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences,*, 207(1167):187–217, February 1980.

[11] Image Metrics and USC Institute for Creative Technologies Graphics Lab. High resolution face scanning for "digital emily". `http://gl.ict.usc.edu/Research/DigitalEmily/`, 2008.

[12] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 101–108, 1994.

[13] Jutta Schumann, Thomas Strothotte, Stefan Laser, and Andreas Raab. Assessing the effect of non-photorealistic rendered images in cad. *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 35–41, 1996.

[14] Adrian Secord. Weighted voronoi stippling. *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43, 2002.

[15] Mario Costa Sousa and Przemyslaw Prusinkiewicz. A few good lines: Suggestive drawing of 3d models. *Computer Graphics Forum*, 22(3):381–390, September 2003.

[16] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to render frames and influence people. *Computer Graphics Forum*, 13(3):455–466, 1994. Special issue on Eurographics '94.

[17] Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, June 2002.

[18] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100, 1994.

[19] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1221–1226, New York, NY, USA, 2006. ACM.