Non-Photorealistic Rendering for Energy Conservation

Yamini Nimmagadda, Yung-Hsiang Lu, Edward J Delp, and David S. Ebert
School of Electrical and Computer Engineering, Purdue University

Abstract

As images are increasingly used in wireless communication, such as mobile phones and PDAs, it is important to reduce the energy consumption for transmitting and receiving images. The energy is approximately proportional to the sizes (numbers of bytes) of the images. Many existing techniques aim to improve compression ratios while preserving the image fidelity without perceivable differences. In this paper, we propose a new approach by allowing visible distortion in the images. Our method eliminates or reduces the fine details (textures) so that new images have smaller file sizes and require less energy to transmit or receive. Even though new images may be visually different, the essential information is preserved. Our experiment uses 400 images and achieves up to 40.1% and average 32.2% reduction in file sizes.

keywords: image processing, energy conservation, non-photorealistic rendering

1 Introduction

The usage of photographs and images is becoming increasingly popular on portable systems, such as mobile phones. These systems usually operate on battery power, so energy conservation is very important. Many studies have been conducted to save energy for multimedia contents on battery-powered systems [1] [2] [3]. [4] [5] [6]. As images constitute a significant portion of the data traffic and storage in these devices, it is important to reduce the energy consumption for transmitting, receiving, and storing the images. The energy to transmit or receive an image is proportional to the sizes (numbers of bits) of the images.

We measure the correlation between the image sizes and reception energy on a PDA. The images reside on a web server and are downloaded to the PDA. The sizes of images in a PDA are usually less than 100 KB. As can be seen in Figure 1, the energy has a strong correlation with the image size. Hence, reducing the image size is an effective way to save energy. The energy is not strictly linear because of the variations of the network condition and the utilization of the web server. The energy needed to transmit an image is also approximately proportional to the image size. The memory space required to store an image is equal to the size of the image. Hence our method focuses on reducing the image size to reduce the energy consumption and memory requirements.

Several studies [1] [2] [3] [4] [5] have been conducted to save energy used for transferring or storing multimedia contents such as images and videos on battery-powered systems. We classify the existing literature into two major categories: image processing techniques and backlight adjustment techniques. The former [1] [2] [3] removes high frequency information without substantially affecting the visual quality. The latter [4] [5] dims the background to reduce the power consumption because representation of dimmed colors requires fewer numbers of bits per pixel. We further classify the existing literature on image processing techniques into two categories namely photorealism and non-photorealism. The former keeps the new image as close to the original image as possible. In contrast, the latter [7] generates or modifies an image and the result is visually different from the original image; however, the new image preserves the significant information in the original image.

In this paper, we present an algorithm for non-photorealistic rendering of images to reduce their sizes and thus the energy for transmission or reception through wireless networks. Our algorithm reduces the

Multimedia on Mobile Devices 2008, edited by Reiner Creutzburg, Jarmo H. Takala
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6821, 682109, © 2008 SPIE-IS&T · 0277-786X/08/$18

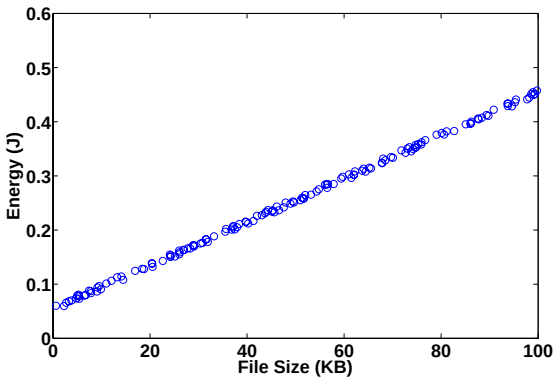
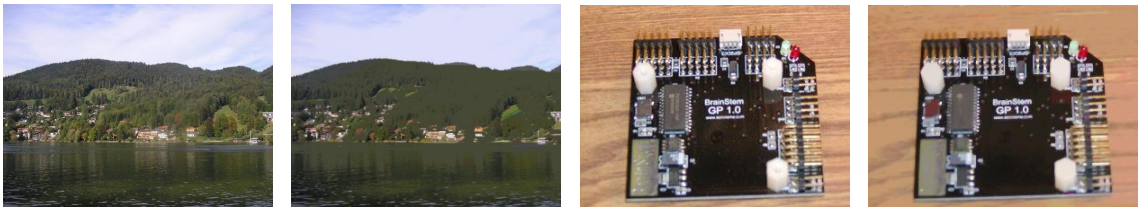


Figure 1: (a) Energy consumption versus image sizes received by a PDA



(a) (b) (c) (d)

Figure 2: (a)(c) Original images (b)(d) processed images (all images are JPEG compressed with resolution 400 × 300). It can be seen that the fine details in clouds, trees, water are removed in (b) and texture on the table and fine details of the circuit components are omitted in (d). The sizes of images are reduced by 34.7% and 28.2% respectively.

image sizes by preserving most of the important information and removing the textures of the constituent objects. The new image is *visually different* but the crucial information is preserved. Figures 2 (a) and (c) are obtained from a digital camera and resized to the resolution 400 × 300 (sizes 15.1 KB and 21.6 KB respectively). Figures 2 (b) and (d) show the processed images. In Figure 2 (b), the clouds in the background and the fine details in tree leaves are removed. The image size is reduced by 33.1% (size 10.1 KB). Figure 2 (d) removes the color variations in the background and makes it appear uniform. This image is 27.8% smaller than the original image (size 15.6 KB).

Our algorithm relies on the capability of human perception for detection, extraction and assimilation of the important information from images at the first glance. Our target system is a hand-held device that receives images from wireless networks for various applications, such as personal photographs or web surfing. Because of small displays in hand-held devices, the textures and fine details cannot be seen clearly; the fine details can be omitted from the images. We develop a new algorithm for detecting continuous edges of the boundaries and omitting the edges of textures. Our experiments use 400 different images and obtain 12.3% to 40.1% reduction in file sizes.

2 Related Work

In the past forty years, the mainstream of computer graphics and image processing was to create or preserve "photorealism". Increasingly complex modeling and shading algorithms are adopted to provide the illusion

that a computer-generated image is as "real" as one seen directly by human eyes. Since mid 1990, a new approach has been developed to *intentionally* abandon the goal of photorealism. This approach is generally called non-photorealistic rendering (NPR) [8] or stylistic rendering. In this section we describe conventional NPR techniques used mainly for artistic purposes. The basic goal of NPR is to use computers to generate or modify images so that they appear to be created by artists, instead of cameras.

Haeberli [9] introduced the technique of operating on source images to obtain artistic NPR effects. This idea has led to the rise of impressionistic painting style: a technique to convert a photographic image into a painting by applying brush strokes of appropriate colors, sizes, shapes and orientations [10] [11]. Other NPR techniques such as pen and ink [12] [13], hatching and shading [14] also operate on source images to create artistic impressions.

Winnemoller et al. [15] presented real-time video abstraction using an NPR method that operates on frames of source images in a video. This technique enhances the high contrast regions and suppresses the low contrast regions, thus preserving significant structural properties of the image frames. Gooch et al. [16] created illustrations from the source images using brightness and luminance thresholding techniques. Gooch discovered that the recognition speed of cartoons or caricatures is the same as that of the original images, while the illustrations require only 8% to 16% storage space. This technique removes or dims the background to reduce the image storage size. In contrast, we present a method that preserves backgrounds and is also applicable to more complex images, such as outdoor photographs that may include buildings, cars, people, trees, sky and clouds, with significant information in the background.

Many studies have been conducted to save transmission or reception energy for images. Irani et al. [4] [5] and Cheng et al. [1] [2] used image processing techniques to enhance contrast and dim backlight. Shim et al. [3] presented a compression algorithm to save energy consumed by display. Mochocki et al. [6] analyzed graphics programs to predict the computation need and reduce power consumption using dynamic voltage scaling. Fox et al. [17] used proxy servers to adjust image resolutions for hand-held devices to fit large images into a smaller display. In this paper, we present an NPR technique for power conservation in portable devices by preserving significant visual information and omitting fine details reducing the image file size and hence, the energy consumption. We believe that our research is the first to employ non-photorealistic rendering for energy conservation on portable devices.

3 Non-photorealistic Method for Energy Conservation

Our goal is to remove the fine details of images and reduce file sizes without losing the visual information. We use 400 test photographs including buildings, cars, people, trees, water (lake or river), sky, and clouds. All images are compressed in the JPEG format. Most of the image processing techniques such as filtering, edge detection, and segmentation are defined for intensity (gray-scaled) images. Hence we separate the color image into monochromatic layers containing only intensity information. We use RGB color model because it is an additive color model representing each color as the linear combination of red, blue and green intensities. The layers of the image are separately processed and recombined at the end to obtain the final color image. Our method is divided into three steps: (1) detect continuous edges using a parameter called "Intensity Gradient Threshold" (IGT), (2) segment the image into regions containing pixels with intensities with a small amount of difference not exceeding IGT, (3) normalize the intensity in each segment.

3.1 Continuous Edge Detection

We eliminate fine details in the image by suppressing boundaries of the details and selectively eliminating them. Traditional edge detection algorithms like Canny and Sobel cannot augment the edges of boundaries and suppress the insignificant details simultaneously. The Canny edge detector finds edges of the subtle details and textures of the constituent objects. The Sobel edge detector finds edges of the boundaries of the objects and ignores the textures. However, Sobel detection results in many discontinuities in the edges

and may lead to overlapping of segments corresponding to different objects with color dispersion from one object to another. The threshold can be tuned manually to produce continuous edge images. However, it is preferred to automatically detect continuous edges; hence, we improve Sobel edge detector. We use a Sobel edge image as a precursor to find the continuous edges of object boundaries. We use a parameter known as *intensity gradient threshold* (IGT) to find this edge image. We define IGT as the minimum intensity difference between consecutive maxima in the pixel intensity histogram. IGT means the maximum intensity difference between highly populated regions in intensity levels. Figure 3 shows an example of an intensity histogram. We find the minimum difference between the local maxima in the histograms of all three colors layers and call it "intensity gradient threshold" (IGT). The IGT in this figure is 21.

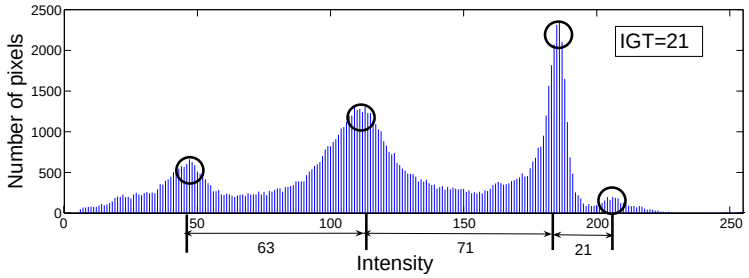


Figure 3: Intensity histogram. The minimum difference between the local maxima is 21 as the value of IGT.

Let function $f(n)$ denote the frequency of the pixel intensity n $(0 \leq n \leq 255)$. This function gives the number of pixels with intensity n. A point n_{max} is a local maximum of the function $f(n)$ if there exists a positive integer ϵ such that $f(n_{max}) \geq f(n)$ for all n with $|n - n_{max}| < \epsilon$. We then define a function $g(n)$ as a train of impulses at the local maxima of $f(n)$:

$$g(n) = \begin{cases} n & \text{for } n = \text{ local maximum of } f(n) \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The intensity gradient threshold is defined as

$$IGT = \min(\{g(n+1) - g(n)\}). \tag{2}$$

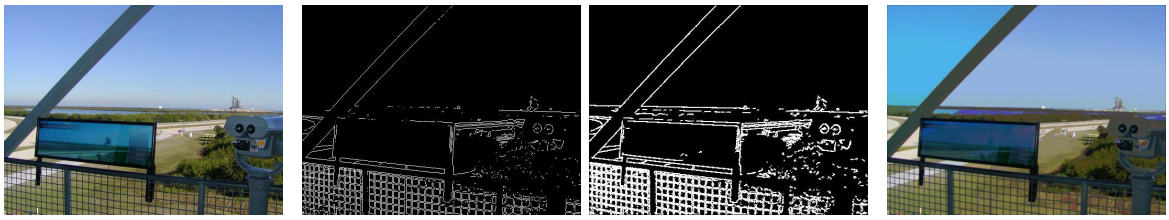
Here $\{g(n+1) - g(n)\}$ represents the set of differences of the intensities corresponding to consecutive local maxima. The gradient is determined at each pixel and the angle of the gradient $G_A(x)$ is computed. The gradient angle is transformed into one of the four directions: east, northeast, north, and southeast. The expression of the gradient angle $G_A(x)$ at a pixel x is given as:

$$G_A(x) = \begin{cases} 0, & -\frac{\pi}{8} \leq G_A(x) < \frac{\pi}{8} \\ \frac{\pi}{4}, & \frac{\pi}{8} \leq G_A(x) < \frac{3\pi}{8} \\ -\frac{\pi}{4}, & -\frac{3\pi}{8} \leq G_A(x) < -\frac{\pi}{8} \\ \frac{\pi}{2}, & \text{otherwise.} \end{cases} \tag{3}$$

There are two immediate neighbors for each pixel along each of the four directions. The neighbors along each direction are shown in the Table 1. We mark a pixel as an edge if the neighbors along the gradient angle are edges. This process joins the discontinuities in the edges. If the intensity difference between the pixels along the direction perpendicular to the gradient angle (shown in fourth column of the Table 1) is greater than the IGT, we mark the pixel as an edge. This step ensures that the edges separating regions of high intensity difference are captured. We obtain a continuous edge image which captures the edges of significant features in the image.

Gradient Angle	Direction	Neighbors	Neighbors in direction perpendicular to gradient angle
0	east	$(x-1,y)(x+1,y)$	$(x,y-1)(x,y+1)$
$\frac{\pi}{4}$	north-east	$(x-1,y-1)(x+1,y+1)$	$(x-1,y+1)(x+1,y-1)$
$-\frac{\pi}{4}$	south-east	$(x-1,y+1)(x+1,y-1)$	$(x-1,y-1)(x+1,y+1)$
$\frac{\pi}{2}$	north	$(x,y-1)(x,y+1)$	$(x-1,y)(x+1,y)$

Table 1: Neighbors of a pixel (x,y) along the four directions



(a) (b) (c) (d)

Figure 4: (a) Original image (19.4 KB), (b) Sobel edge image, (c) Edge image determined by our method, (d) New image (15.3 KB). The reduction in size is 21.1%.

Figures 4 compares our algorithm with the result from Sobel edge detector. Figure 4(a) is the original image. In Figure 4(b), Sobel detector produces discontinuous edges, especially for the railings at the bottom of the image. Discontinuous edges will create many small segments in the next step and limits the reduction of file sizes and energy consumption. Our method generates continuous edges as shown in Figure 4(c). The final processed image is shown the Figure 4(d).

3.2 Image Segmentation

This step enhances the significant properties and suppressing the subtle details of the constituent objects. Each object in the image has a characteristic texture to be treated differently from the other objects. We cluster the pixels of a constituent object into a segment based on the edge image determined in the previous step. We use a bottom-up region-merging approach that can be divided into two phases: initialization and segment merging.

Initialization labels each pixel with a unique number in raster order. These labels are the segments to which the pixels belong; each pixel represents a segment. Let I be the image as the union of all the segments $I_i, 0 < i \leq N$ (N is the number of pixels).

$$I = \bigcup_{i=1}^{N} I_i. \tag{4}$$

Segment merging computes the mean of the intensity of all the pixels in a segment, called mean intensity M_i for segment I_i. If the difference between means M_i and M_j of two adjacent segments is less than IGT, the two segments I_i and I_j are merged into a single segment I_i where $M_i < M_j$. All the elements are removed from I_j. All the pixels of the segment I_j take the label i after merging.

$$I_i = I_i \cup I_j \quad \text{and} \quad I_j = \phi, \qquad \text{if} \quad (M_i < M_j). \tag{5}$$

This process continues until the intensity differences of all adjacent segments are larger than IGT.

3.3 Intensity Normalization

In this step, we store the mean intensity information of a segment and compare it with that of the neighboring segments. If the difference is less than the *frequency threshold* (FT), we merge these two segments. FT is the minimum difference between the values of adjacent local maxima in the intensity histogram. IGT and FT are calculated from the same points in the intensity histogram, but IGT is computed on intensity axis where as FT is computed on frequency axis. Let $f(n)$ denote the frequency function described in Section 3.1. We define a function $h(n)$ such that

$$h(n) = \begin{cases} f(n), & n \text{ is the local maximum of } f(n) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The frequency threshold is defined as

$$FT = \max(\{h(n+1) - h(n)\}). \tag{7}$$

Initially the mean intensity of the first segment is assigned a variable α. If the difference between the mean intensity of a segment and α is greater than FT, the mean intensity of this segment is set to α. If the intensity of a pixel in a segment exceeds the mean intensity, the intensity of the pixel is set to the mean intensity. This normalization process removes details by changing the various intensity levels to the mean intensity.

The red, blue, and green layers are recombined into a single color image. The processed image can be visually different from the original image. All the prominent structural properties are preserved in the image and only the fine details are removed.

4 Implementation

We implement our algorithm in MATLAB. The original and the processed images are transmitted to a PDA-like system for measuring the energy consumed for reception. This system is an Integrated Development Platform (IDP) from Accelent Systems, Inc. It uses similar components (such as an XScale processor) that can be found in many hand-held devices. IDP can run Windows or Linux (2.4.18 in our setup); IDP has many probing points through which we can measure the power consumption of different components. The images are stored on a web server and retrieved to IDP using the `wget` program. We measure the power consumption of an Orinoco wireless card using a National Instruments' data acquisition card.

We used 400 different images in our experiments. The experimental results indicate that our method is robust across different types of images. Some images contain dominant objects and some do not. The images are taken in varying weather conditions. The distributions of the size reduction is shown in Figure 5. Most images can achieve 30% to 35% reduction in file sizes. Images without significant textures have fewer fine details, so the size cannot be further reduced using our method. Images with textures have more details; hence they achieve better size reduction ($> 35\%$). The two extreme cases are shown in Figure 6: (a) and (c) are the original images, (b) and (d) are the processed images with size reductions 12.3% and 40.1% respectively.

5 Comparison with Other Methods

In this section, we compare our method with other image processing techniques, including edge images, illustrations, low pass filtered images, anisotropic filtered images.

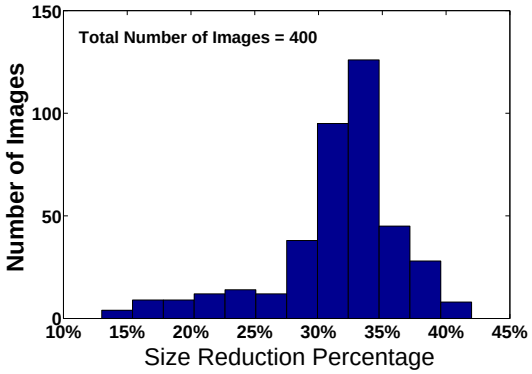
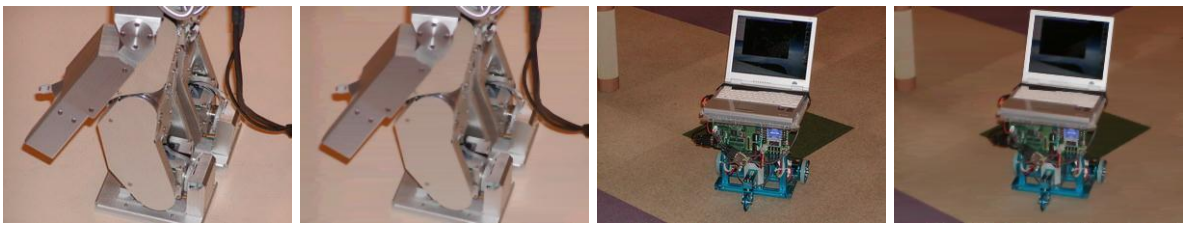


Figure 5: Distribution of size reduction. Most images can obtain 30-35% size reductions. The images with size reduction above 35% are highly textured and those below 25% are not highly textured.



(a) Original Image (b) Processed Image (c) Original Image (d) Processed Image

Figure 6: (a) original image (size: 9.29 KB) (b) processed image (size: 8.15 KB). The reduction in size is 12.3%. The reduction is small because there is no visible texture in the image. (c) original image (size: 10.8 KB) (d) processed image (size: 6.47 KB). The reduction in size is 40.1%. The texture of the floor causes the reduction to be much higher.

5.1 Edge Detection

Edges occur due to the discontinuities as the boundaries between distinct objects. The Canny edge detection algorithm [18] can detect the edges missed in the Sobel algorithm. The Canny detector first smoothens the image to eliminate high frequency components. It then finds the image gradient (slope) to highlight regions with high intensity differences. The algorithm tracks along these regions and suppresses any pixel that is not at the maximum. The gradients are further reduced by hysteresis using two thresholds. If the magnitude is below the first threshold, the pixel is set to zero (not an edge pixel). If the magnitude is above the high threshold, it is made an edge. If the magnitude is between the two thresholds, it is made an edge if the neighbors are edges.

5.2 Low Pass Filtering

A low pass filter [18] allows low frequency data that does not change much from a pixel to neighboring pixels and removes the high frequencies. The new image appears blurred or smoothened. For an image that contains a high variation in colors, such a filter smoothens the image and reduces the noise with negligible effects on important features. The low pass filter used in this comparison is defined as follows:

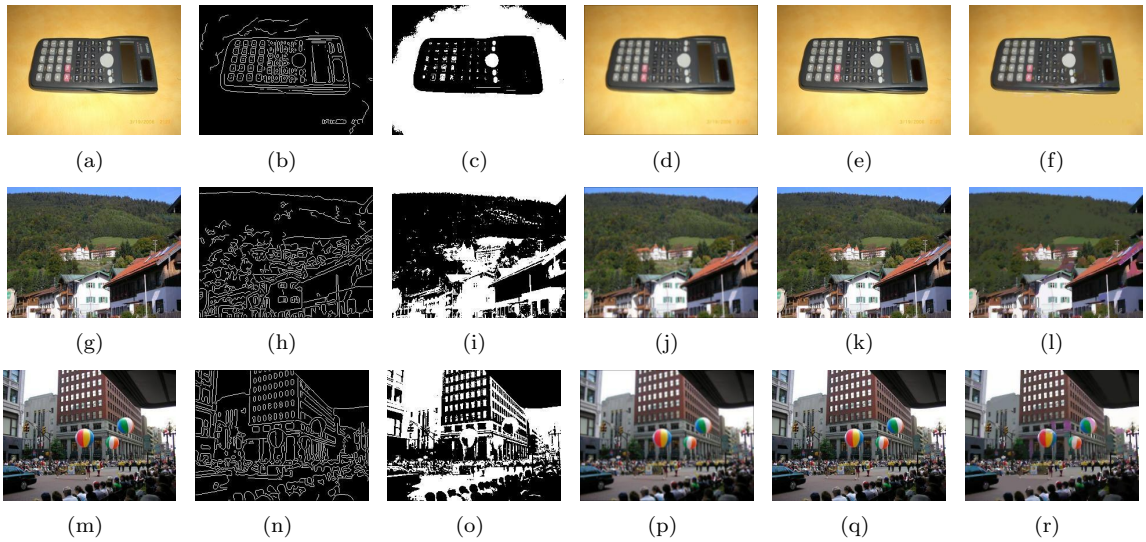


Figure 7: Comparisons of different methods. (a)(g)(m) original photographs, (b)(h)(n) Canny edge images, (c)(i)(o) illustrations, (d)(j)(p) low pass filtered images, (e)(k)(q) anisotropic filtered images, (f)(l)(r) images processed by our method.

$$h_{lp} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{8}$$

5.3 Illustrations

Gooch's illustrations [16] are two-tone images produced by thresholding the gray scale images. The illustrations are produced by operating on brightness and luminance; thus, we convert the color images to gray scaled images using the following transformation [18]:

$$Y = 0.299R + 0.587G + 0.114B, \tag{9}$$

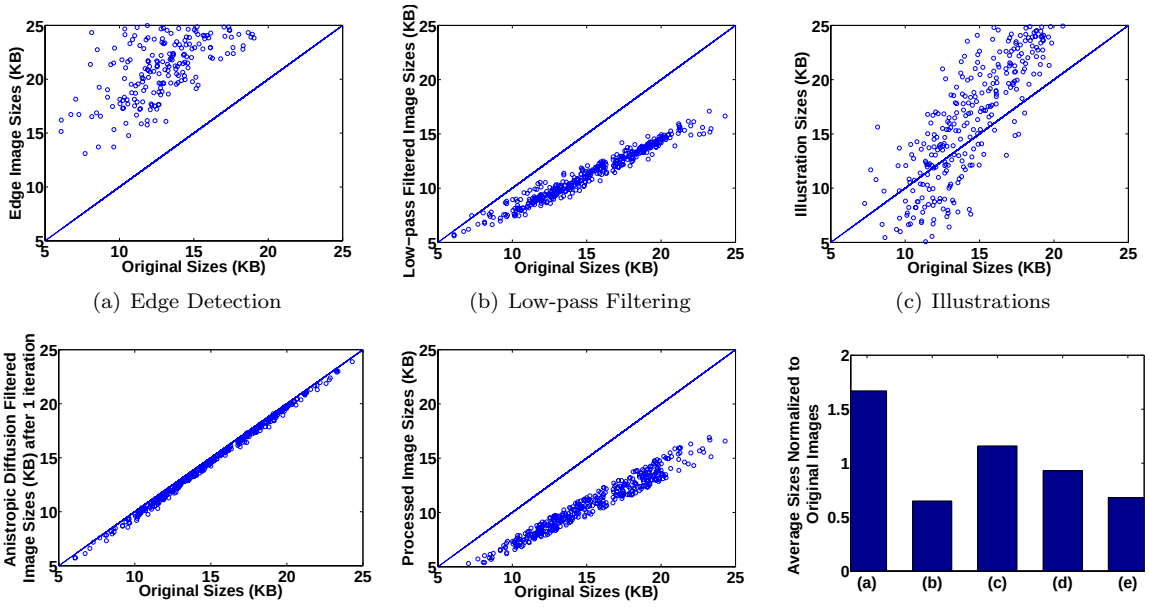
where Y is the gray scale component in the YIQ color space used in NTSC television and R, G, B are the tristimulus values of the color images in RGB color space. The illustrations preserve the key information and remove unnecessary information from the images. However, sometimes illustrations miss important information particularly in outdoor scenes, when the background can be important.

5.4 Anisotropic Diffusion Filtering

Anisotropic diffusion filters [19] are iterative and tunable filters that enhance the contrast and facilitate easy edge detection. These filters have two Gaussian kernels, one operating in the image domain (pixel-positions) and the other operating in the image range (intensity values at the pixels). The anisotropic diffusion transformation of the intensity of the image at a pixel $s = (x, y)$ with a neighborhood $N(s)$ is shown in equation:

$$I(s) = \frac{\sum_{\hat{s}\epsilon N(s)} G_d(s,\hat{s}) G_r(s,\hat{s}) I(\hat{s})}{\sum_{\hat{s}\epsilon N(s)} G_d(s,\hat{s}) G_r(s,\hat{s})}, \tag{10}$$

where $G_d(s,\hat{s})$ is the Gaussian kernel on the domain with variance σ_d^2 and $G_r(s,\hat{s})$ is the Gaussian kernel on the range with variance σ_r^2 of the intensity feature space defined by:



(a) Edge Detection (b) Low-pass Filtering (c) Illustrations

(d) Anisotropic Filtering (e) Our Method (f) Average Sizes

Figure 8: (a),(b),(c),(d),(e) sizes of processed images versus the sizes of original images. It can be seen that the processed images lie in the region $y < x$ for (b), (d), and (e) . (f) compares average sizes normalized to original images for various methods

$$G_d(s, \hat{s}) = e^{\frac{-1}{2}\frac{\|s-\hat{s}\|}{\sigma_d}^2}, \tag{11}$$

$$G_r(s, \hat{s}) = e^{\frac{-1}{2}\frac{\|I(s)-I(\hat{s})\|}{\sigma_r}^2}. \tag{12}$$

Figure 7 shows the images processed using different techniques. Our test images include different types like portraits, outdoor scenes, outdoor scenes with people, and images with significant objects. Figure 8 shows the sizes of processed images versus original images for various methods. The y-axis of the figure represents processed image size and the x-axis represents the original image size. The results indicate that our method is applicable to different types of images. As can be seen from Figure 8 (b), (d) and (e), low-pass filtering, anisotropic diffusion filtering and our method always reduce the image sizes. Edge images and illustrations are not always smaller than the original images. Table 2 compares several aspects of these methods. The images processed by our method are almost as small as the low-pass filtered images but also preserve the sharpness of edges.

We conducted an on-line survey to compare the visual qualities of images processed using low-pass filtering, anisotropic diffusion filtering, and our method. Illustrations and edge images are visually different from the original images, so they are not considered in this survey. The users were asked to rank (1 - very close to the original image, 2 - moderately close to the original image, 3 - not close to the original image) the processed images based on the resemblance to original images. Based on 64 responses, it is observed that the visual quality of images processed by our method is higher than low-pass filtering and lower than anisotropic filtering.

6 Conclusion

We present an image processing algorithm to reduce the sizes of photographs and hence the energy to receive them on battery-powered portable systems. Our method removes fine details in photographs. Even though

Smoothing Technique	Processed image is always smaller than the original	Output is a color image	Preserves sharpness of edges	Average size normalized to original image	Visual quality (from on-line survey)
Edge Detection	No	No	Yes	1.67	NA
Low pass filter	Yes	Yes	No	0.65	2.73
Illustrations	No	No	Yes	1.16	NA
Anisotropic filter	Yes	Yes	Yes	0.93	1.49
Our method	Yes	Yes	Yes	0.68	1.63

Table 2: Comparison of various techniques

the new images lose photorealism and sometimes appear like paintings, all crucial information is preserved. We achieve 12.3% to 40.1% reduction in file sizes. We compare the image sizes and visual qualities of the images modified by different methods.

Acknowledgments: This project is supported in part by NSF CCF-0541267. Any opinions, findings, and conclusions or recommendations in the projects are those of the investigators and do not necessarily reflect the views of the sponsors. We appreciate the help from Joshus M Speciale.

References

[1] Wei-Chung Cheng and Chain-Fu Chao. Minimization for LED-backlit TFT-LCDs. In *Design Automation Conference*, pages 608–611, 2006.

[2] Wei-Chung Cheng, Chih-Fu Hsu, and Chain-Fu Chao. Temporal Vision-Guided Energy Minimization for Portable Displays. In *International Symposium on Low Power Electronics and Design*, pages 89–94, 2006.

[3] Hojun Shim, Naehyuck Chang, and Massoud Pedram. A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems. In *Asia South Pacific Design Automation Conference*, pages 818–823, 2004.

[4] Ali Iranli, Hanif Fatemi, and Massoud Pedram. HEBS: Histogram Equalization for Backlight Scaling. In *Design Automation and Test in Europe*, pages 346–351, 2005.

[5] Ali Iranli, Wonbok Lee, and Massoud Pedram. Backlight Dimming in Power-Aware Mobile Displays. In *Design Automation Conference*, pages 604–607, 2006.

[6] Bren C Mochocki, Kanishka Lahiri, Srihari Cadambi, and X Sharon Hu. Signature-Based Workload Estimation for Mobile 3D Graphics. In *Design Automation Conference*, pages 592–597, 2006.

[7] Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann, 2002.

[8] John Lansdown and Simon Schofield. Expressive Rendering: A Review of Nonphotorealistic Techniques. *IEEE Computer Graphics and Applications*, 15(3):29–37, May 1995.

[9] Paul Haeberli. Paint by Numbers: Abstract Image Representations. In *SIGGRAPH*, pages 207–214, 1990.

[10] Aaron Hertzmann. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *SIGGRAPH*, pages 453–460, 1998.

[11] Michael Haller and Daniel Sperl. Real-Time Painterly Rendering for MR Applications. In *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 30–38, 2004.

[12] Michael P Salisbury, Michael T Wong, John F Hughes, and David H Salesin. Orientable Textures for Image-Based Pen-and-Ink Illustration. In *SIGGRAPH*, pages 401–406, 1997.

[13] Georges Winkenbach and David H Salesin. Rendering Parametric Surfaces in Pen and Ink. In *SIGGRAPH*, pages 469–476, 1996.

[14] Pierre-Marc Jodoin, Emric Epstein, Martin Granger-Piche, and Victor Ostromoukhov. Hatching by Example: a Statistical Approach. In *International Symposium on Non-Photorealistic Animation and Rendering*, pages 29–36, 2002.

[15] Holger Winnemoller, Sven C Olsen, and Bruce Gooch. Real-Time Video Abstraction. In *ACM SIGGRAPH*, pages 1221–1226, 2006.

[16] Bruce Gooch, Erik Reinhard, and Amy Gooch. Human Facial Illustrations: Creation and Psychophysical Evaluation. *ACM Transactions on Graphics*, 23(1):27–44, January 2004.

[17] Armando Fox, Steven D Gribble, Yatin Chawathe, and Eric Brewer. Adapting To Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives. *IEEE Personal Communications*, 5(4):10–19, August 1998.

[18] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[19] Pietro Perona and Jitendra Malik. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.