Bilateral filtering in images & its applications

Aryan Agal
(16D170004)
Satyaprajna Sarthak Sahoo
(16D170026)
Department of Energy Science and Engineering
Indian Institute of Technology, Bombay
Mumbai, India

***Abstract*—This project explores bilateral filtering in images and implements two practical applications- in digital flash photography and in cartoonification of images. We have implemented the paper entitled "Digital Photography with Flash and No-Flash Image Pairs" Petschnigg et al. We then devised an algorithm for toonification and implemented it.**

Keywords*— *Bilateral filtering, shadow and specularity detection, flash to ambient detail transfer, cartoonification.

I. Introduction

The bilateral filter is technique to smooth images while preserving edges. It can be traced back to 1995 with the work of Aurich and Weule on nonlinear Gaussian filters. It has been later rediscovered by Smith and Brady as part of their SUSAN framework, and Tomasi and Manduchi who gave it its current name. Since then, the use of bilateral filtering has grown rapidly and is now ubiquitous in image-processing applications. It has been used in various contexts such as denoising, texture editing and relighting , tone management, demosaicking, stylization, and optical-flow estimation. The bilateral filter has several qualities that explain its success:

- Its formulation is simple: each pixel is replaced by an average of its neighbors. This aspect is important because it makes it easy to acquire intuition about its behavior, to adapt it to application-specific requirements, and to implement it.
- It depends only on two parameters that indicate the size and contrast of the features to preserve
- It can be used in a non-iterative manner. This makes the parameters easy to set since their effect is not cumulative over several iterations.

In parallel to applications, a wealth of theoretical studies explained and characterized the bilateral filter's behavior. The strengths and limitations of bilateral filtering are now fairly well understood. As a consequence, several extensions have been proposed.

II. Preliminaries

To introduce bilateral filtering, we first describe in Section the Gaussian convolution. This filter is close to the bilateral filter but is not edge-preserving. We will use it to introduce the notion of local average and to underscore the specificities of the bilateral filter that make it edge-preserving.

A. Linear Filtering with Gaussian Blur (GB)

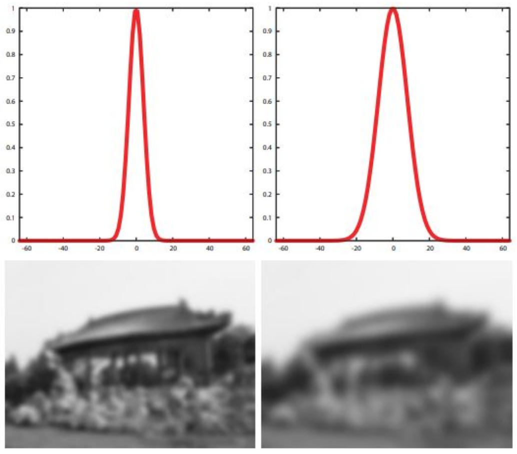
Convolution by a positive kernel is the basic operation in linear image filtering. It amounts to estimate at each position a local average of intensities and corresponds to low-pass filtering. One defines the Gaussian blur (GB) filtered image by:

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_\sigma(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

where $G_\sigma(x)$ denotes the two-dimensional Gaussian kernel:

$$G_\sigma(x) = \frac{1}{2\pi\,\sigma^2}\,\exp\left(-\frac{x^2}{2\,\sigma^2}\right)$$

So, Gaussian filtering is a weighted average of the intensity of the adjacent positions with a weight decreasing with the spatial distance to the center position p. This distance is defined by $G\sigma(\|p - q\|)$, where σ is a parameter defining the extension of the neighborhood. As a result, image edges are blurred as shown.


$\sigma = 4$ $\sigma = 8$

B. Nonlinear Filtering with Bilateral Filter (BF)

Similarly to the Gaussian convolution, the bilateral filter is also defined as a weighted average of pixels. The difference is that the bilateral filter takes into account the variation of intensities to preserve edges. The rationale of bilateral filtering is that two pixels are close to each other not only if they occupy nearby spatial locations but also if they have some similarity in the photometric range. The formalization of this idea goes back in the literature to Yaroslavsky [1985], then Aurich and Weule [1995], Smith and Brady [1997b] and Tomasi and Manduchi [1998]. The bilateral filter, denoted by BF[.], is defined by:

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})\, I_{\mathbf{q}}$$

where Wp is a normalization factor:

$$W_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})$$

Parameters σ_s and σ_r will measure the amount of filtering for the image I. Equation (3) is a normalized weighted average where Gσs is a spatial Gaussian that decreases the influence of distant pixels, Gσr a range Gaussian that decreases the influence of pixels q with an intensity value different from Ip. Note that the term range qualifies quantities related to pixel values, by opposition to space which refers to pixel location. Figure 4 shows a sample output of the bilateral filter and Figure 5 illustrates how the weights are computed on a simple example

Lena Test Image Smoothened by bilateral filter

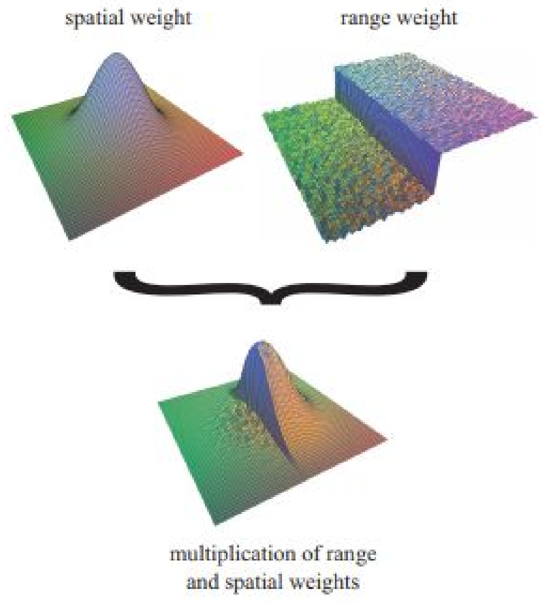


Illustration of Multiplication of weights

C. *Joint bilateral filter*

We modify the basic bilateral filter to compute the edge-stopping function g_r using the flash image F instead of A. We call this technique the joint bilateral filter.

$$A_p^{NR} = \frac{1}{k(p)} \sum_{p'\in\Omega} g_d(p'-p)\, g_r(F_p - F_{p'})\, A_{p'},$$

where *k(p)* is modified similarly. Here A^{NR} is the noise-reduced version of A . We set σ_d just as we did for the basic bilateral filter. Under the assumption that F has little noise, we can set σ_r to be very small and still ensure that the edge-stopping function $g_r(F_p - F_{p'})$ will choose the proper weights for nearby pixels and therefore will not over-blur or under-blur the ambient image.

III. Digital Photography with Flash and No-Flash Image Pairs: Paper

An important goal of photography is to capture and reproduce the visual richness of a real environment. Lighting is an integral aspect of this visual richness and often sets the mood or atmosphere in the photograph. As photographers know, however, the use of flash can also have a negative impact on the lighting characteristics of the environment. Objects near the camera are disproportionately brightened, and the mood evoked by ambient illumination may be destroyed. In addition, the flash may introduce unwanted artifacts such as red eye, harsh shadows, and specularities, none of which are part of the natural scene.

Today, digital photography makes it fast, easy, and economical to take a pair of images of low-light environments: one with flash to capture detail and one without flash to capture ambient illumination. (We sometimes refer to the no-flash image as the ambient image.) In this paper, the authors present a variety of techniques that analyze and combine features from the images in such a flash/no-flash pair:

A. *Denoising*

Reducing noise in photographic images has been a long-standing problem in image processing and computer vision. The bilateral filter is a fast, non-iterative technique, and has been applied to a variety of problems beyond image denoising, including tonemapping [Durand and Dorsey 2002; Choudhury and Tumblin 2003], separating illumination from texture [Oh et al. 2001] and mesh smoothing [Fleishman et al. 2003; Jones et al. 2003].

The paper's ambient image denoising technique also builds on the bilateral filter and the joint bilateral filter.

In practice, the authors have found that σ_r can be set to 0.1% of the total range of color values. Unlike basic bilateral filtering, we fix σ_r for all images. The joint bilateral filter relies on the flash image as an estimator of the ambient image. Therefore it can fail in flash shadows and specularities because they only appear in the flash image. At the edges of such regions, the joint bilateral filter may under-blur the ambient image since it will down-weight pixels where the filter straddles these edges. Similarly, inside these regions, it may overblur the ambient image. This problem is solved by first detecting flash shadows and specular regions and then falling back to basic bilateral filtering within these regions.

The joint bilateral filter relies on the flash image as an estimator of the ambient image. Therefore it can fail in flash shadows and specularities because they only appear in the flash image. At the edges of such regions, the joint bilateral filter may under-blur the ambient image since it will down-weight pixels where the filter straddles these edges. Similarly, inside these regions, it may overblur the ambient image.

$$A^{NR'} = (1-M)\,A^{NR} + MA^{Base}\ .$$

B. *Flash-To-Ambient Detail Transfer*

While the joint bilateral filter can reduce noise, it cannot add detail that may be present in the flash image. Yet, as described in Section 2, the higher SNR of the flash image allows it to retain nuances that are overwhelmed by noise in the ambient image. Moreover, the flash typically provides strong directional lighting that can reveal additional surface detail that is not visible in more uniform ambient lighting. The flash may also illuminate detail in regions that are in shadow in the ambient image. To transfer this detail we begin by computing a detail layer from the flash image as the following ratio:

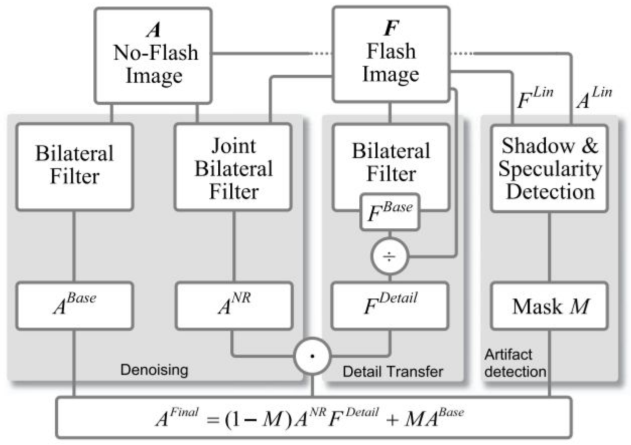
$$F^{Detail} = \frac{F + \varepsilon}{F^{Base} + \varepsilon}$$

where Base *F* is computed using the basic bilateral filter on *F.* The ratio is computed on each RGB channel separately and is independent of the signal magnitude and surface reflectance. The ratio captures the local detail variation in *F* and is commonly called a quotient image or ratio image in computer vision. The advantage of using the bilateral filter to compute Base F rather than a classic low-pass Gaussian filter is that we reduce haloing. At low signal values, the flash image contains noise that can generate spurious detail. We add ε to both the numerator and denominator of the ratio to reject these low signal values and thereby reduce such artifacts (and also avoid division by zero). In practice we use ε = 0.02 across all our results. To transfer the detail, we simply multiply the noise-reduced ambient image A^{NR} by the ratio Detail F . Just as in joint bilateral filtering, our transfer algorithm produces a poor detail estimate in shadows and specular regions caused by the flash. Therefore, we again rely on the detection algorithm to estimate a mask M identifying these regions and compute the final image as:

$$A^{Final} = (1 - M)A^{NR}F^{Detail} + MA^{Base}$$

With this detail transfer approach, we can control the amount of detail transferred by choosing appropriate settings for the bilateral filter parameters σ d andσ r used to create Base F . As we increase these filter widths, we generate increasingly smoother versions of Base F and as a result capture more detail in Detail F . However, with excessive smoothing, the bilateral filter essentially reduces to a Gaussian filter and leads to haloing artifacts in the final image.
In most cases, the detail transfer algorithm improves the appearance of the ambient image. However, it is important to note that the flash image may contain detail that looks unnatural when transferred to the ambient image. For example, if the light from the flash strikes a surfaces at a shallow angle, the flash image may pick up surface texture (i.e. wood grain, stucco, etc.) as detail. If this texture is not visible in the original ambient image, it may look odd. Similarly if the flash image washes out detail, the ambient image may be over-blurred. The paper's approach allows the user to control how much detail is transferred over the entire image. Automatically adjusting the amount of local detail transferred is an area for future work.

The basic summary of the algorithm used in the project is given below:



Finding smooth continuous contours is an important component to achieving the overall effect. The following steps are taken to provide contour detection that works in an artistically pleasing manner on a wide variety of images. All edge processing tasks are performed with a single-channel grayscale image derived from the luminance values of the input.

C. *Detecting Flash Shadows and Specularities*

Light from the flash can introduce shadows and specularities into the flash image. Within flash shadows, the image may be as dim as the ambient image and therefore suffer from noise. Similarly, within specular reflections, the flash image may be saturated and lose detail. Moreover, the boundaries of both these regions may form high-frequency edges that do not exist in the ambient image. To avoid using information from the flash image in these regions, we first detect the flash shadows and specularities. Flash Shadows. Since a point in a flash shadow is not illuminated by the flash, it should appear exactly as it appears in the ambient image. Ideally, we could linearize *A* and *F* as described in Section 3 and then detect pixels where the luminance of the difference image *FLin* − *ALin* is zero. In practice, this approach is confounded by four issues: 1) surfaces that do not reflect any light (i.e. with zero albedo) are detected as shadows; 2) distant surfaces not reached by the flash are detected as shadows; 3) noise causes nonzero values within shadows; and 4) inter-reflection of light from the flash causes non-zero values within the shadow. The first two issues do not cause a problem since the results are the same in both the ambient and flash images and thus whichever image is chosen will give the same result. To deal with noise and inter-reflection, we add a threshold when computing the shadow mask by looking for pixels in which the difference between the linearized flash and ambient images is small:

$$M^{Shad} = \begin{cases} 1 \text{ when } F^{Lin} - A^{Lin} \leq \tau_{Shad} \\ 0 \text{ otherwise.} \end{cases}$$

Noise can contaminate the shadow mask with small speckles, holes and ragged edges. We clean up the shadow mask using image morphological operations to erode the speckles and fill the holes. To produce a conservative estimate that fully covers the shadow region, we then dilate the mask. Flash Specularities. We detect specular regions caused by the flash using a simple physically motivated heuristic. Specular regions should be bright in *FLin* and should therefore saturate the image sensor. Hence, we look for luminance values in the

flash image that are greater than 95% of the range of sensor output values. We clean, fill holes, and dilate the specular mask just as we did for the shadow mask.
Final Merge: We form our final mask *M* by taking the union of the shadow and specular masks. We then blur the mask to feather its edges and prevent visible seams when the mask is used to combine regions from different images.

D. Continuous Flash Adjustment

When taking a flash image, the intensity of the flash can sometimes be too bright, saturating a nearby object, or it can be too dim, leaving mid-distance objects under-exposed. With a flash and non-flash image pair, we can let the user adjust the flash intensity after the picture has been taken. We have explored several ways of interpolating the ambient and flash images. The most effective scheme is to convert the original flash/no-flash pair into YCbCr space and then linearly interpolate them using:

$$F^{Adjusted} = (1-\alpha)A + (\alpha)F$$

To provide more user control, we allow extrapolation by letting the parameter α go outside the normal [0,1] range. However, we only extrapolate the Y channel, and restrict the Cb and Cr channel interpolations to their extrema in the two original images, to prevent excessive distortion of the hue.

IV. Toonification

A. Median Filter

Before any further processing, a median filter is applied in order to reduce any salt and pepper noise that may be in the image. At this stage, the median filter kernel is a 7×7 matrix with the centroid as the center element in the matrix. The RGB pixel values at the centroid are set to be the median of the 49 RGB pixel values in the neighborhood. In this way, any extreme specks are smoothed over. The median filter is small enough to preserve edges in the image, which is important. Too large a kernel size would result in washed out edges.

B. Edge Detection

Edge detection in cartoonification is done using the Sobel operator. This operator performs a 2-D spatial gradient measurement on an image and emphasizes regions of high spatial gradient that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input greyscale image. Compared to other edge operator, Sobel has two main advantages:

- Since the introduction of the average factor,it has some smoothing effect to the random noise of the image.
- Because it is the differential of two rows or two columns,so the elements of the edge on both sides has been enhanced,so that the edge seems thick and bright.

C. Morphological Operations

Currently, the only morphological operation employed by the algorithm at this stage is skeletonisation. The purpose of this step is to make the edge lines after the Sobel processing step. Different combinations of morphological operators are still being tested in this stage, and the aesthetic of the contours in the final image depend largely on this stage

D. Edge Filter

Finally, the edge image is separated into it's constituent regions, and any region with an area below a certain threshold is removed. In this way, small contours picked up by the Canny edge detector are ignored in the final image, which helps reduce unwanted line clutter in the result. Empirical testing with the minimum area threshold showed that a minimum value of 10 seemed to produce the most consistently desirable results across the sample images.

Algorithm for Color operations are:

A. Bilateral Filter

This filter is the key element in the color image processing chain, as it homogenizes color regions while preserving edges, even over multiple iterations. Because it is a computationally expensive task, the image is downsampled by a factor of 4 in both the x and y directions before being filtered. The bilateral filter works similarly to a Gaussian filter in that it assigns to each pixel a weighted sum of the pixel values in the neighborhood. However, the difference is that the weights are further adjusted depending on how different the pixel values are. In this way, a pixel that is close in color to the centroid pixel will have a higher weight than a pixel at the same distance with a more distinct color. This extra step in the weight calculation is important because it means that sharp changes in color (edges) can be preserved, unlike with a simple Gaussian blur. However, the bilateral filter runtime is dependent on the kernel size, and testing showed that running more iterations with a smaller kernel yielded results that were more aesthetically pleasing, and faster than those yielded by applying a bilateral filter with a large kernel size with less iterations. The Toonify algorithm uses a 9×9 kernel and iterates the filter 14 times on the image. Once the filtering is complete, the image is restored to it's original size using linear interpolation to fill-in the missing pixels.

B. Color quantisation

The final step in the color image processing chain is to re-quantize the color palette of the image. This is achieved with each color channel by applying the following formula, where p is the pixel value, and a is the factor by which the number of colors in each channel is to be reduced:

$$p_{new} = \lfloor \frac{p}{a} \rfloor \times a$$

Admittedly, this has the unwanted effect of truncating colors with pixel values close to the maximum, but in practice the absence of max-value colors is hardly noticeable. In this algorithm, a scale of a = 24 was chosen. This means that the standard RGB color palette of 256^3 colors is reduced to a palette of $[\ 256/24\ ]^3$ unique colors. Upon testing, this scale factor produced the most desirable results.

C. Recombine

Once both the color and edge image processing chains are complete, the only task left is to overlay the edges onto the color image. It is possible to draw the edges in different colors on the image, and during development an algorithm was tested wherein each edge is drawn with a color based on its immediate surroundings, but this yielded unpredictable and aesthetically unappealing results. The final algorithm simply draws on all the contours in black.

V. RESULTS

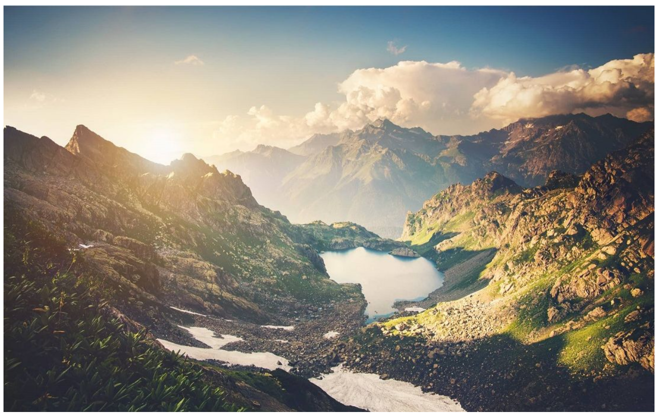
Testing was done on multiple images.

A. Selection of Images

The images selected for cartoonification are:

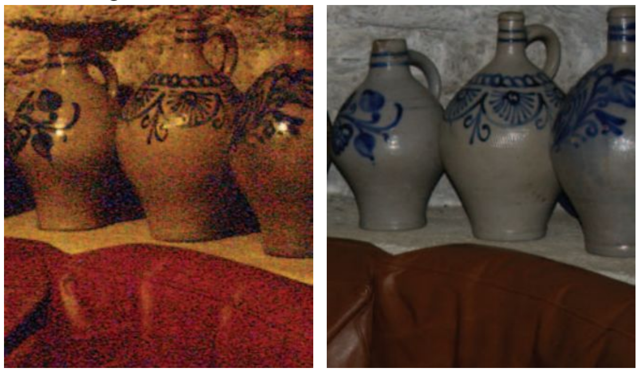
This is a very simple image with very less gradient in colors and simple shades and clearly defined edges. Basically a bilateral transform would not change this image by a large margin.

This image was elected because of the two types of edges. One is the soft edge of the grass blades, while the second is a strong edge of the tiger stripes. This image was selected to test the behaviour of the edge detection method in the two cases.



This is a typical natural image of a landscape with a large type and variety of edges and hence it was chosen to see which edges were chosen.

This is a typical human face and hence it was chosen to see what effect our code had on the natural contours of a human face.
Other images used are as follows:

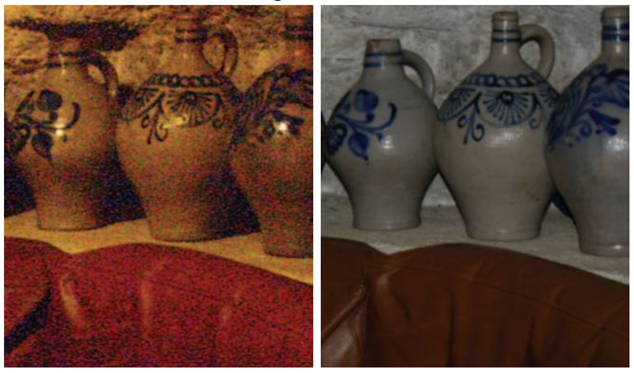


This is the classic example of the flash-no flash pair on which we had to apply our algorithm.

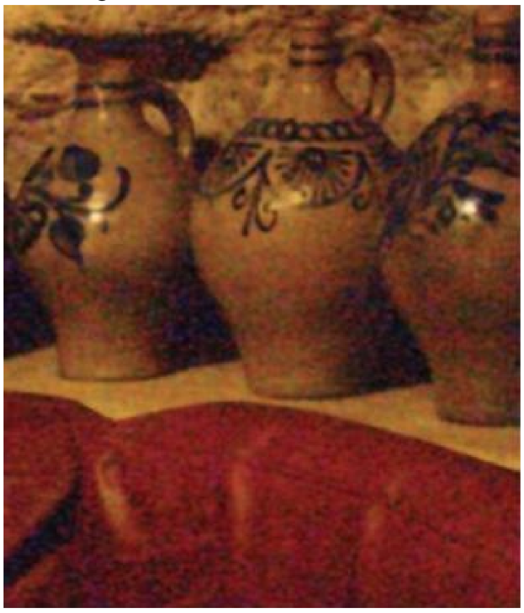
For implementation, large image sizes could not be used, as a bilateral transform in it's raw state is a very expensive process. Hence a resized version of selected images were used.

B. Results obtained

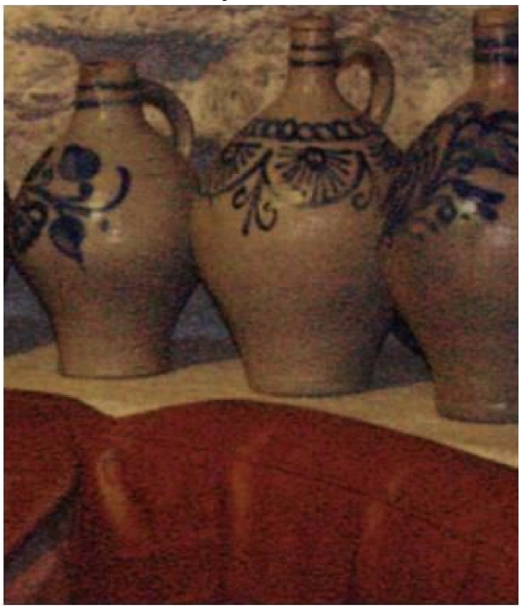
Different tests on the images were done. Here are the results:



Denoising

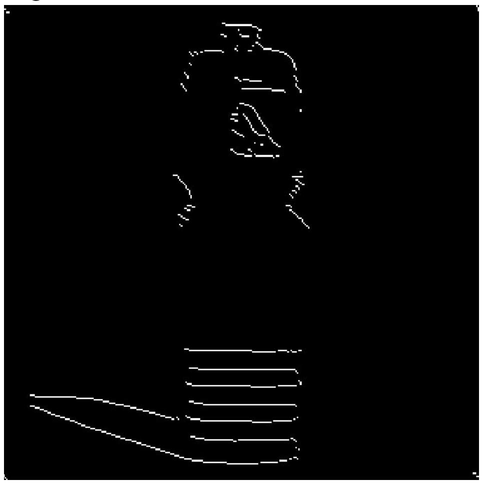


Continuous Flash Adjustment



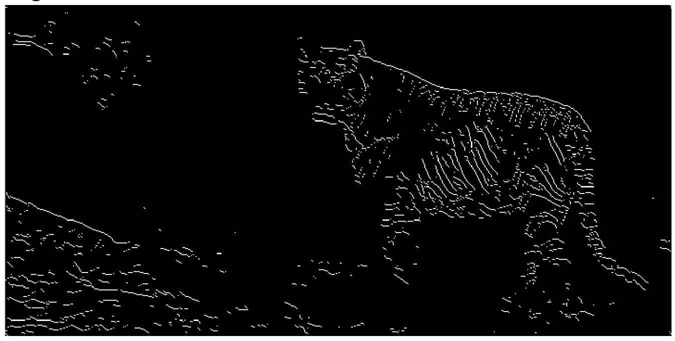
Bottle:

Edge Detection:

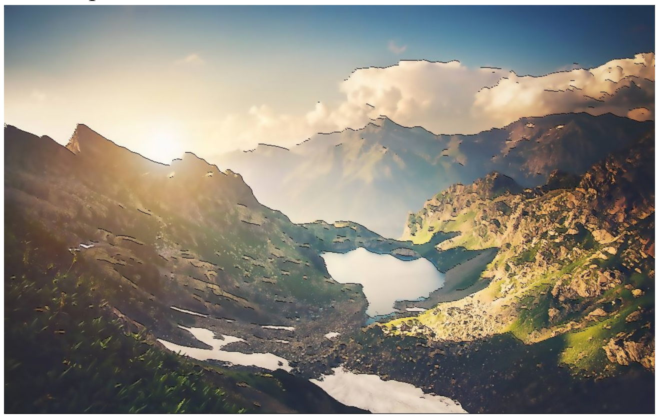


Tiger

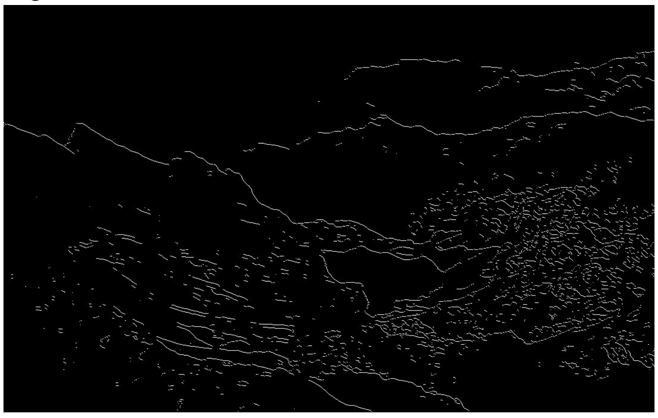
Edge Detection**:**



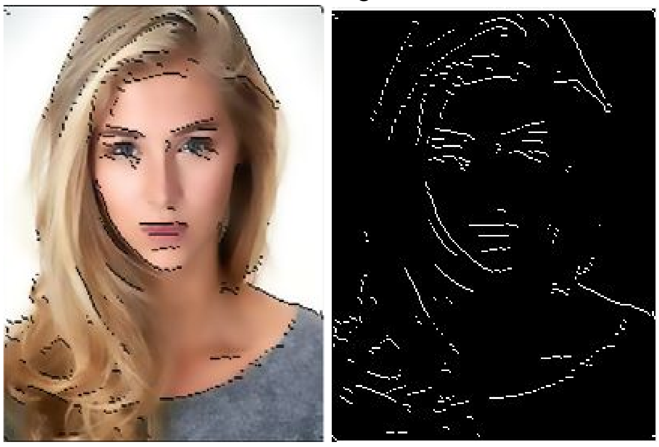
Landscape:



Edge Detection:



Potrait: Edge Detection



We see that the algorithm is particularly bad at handling portrait images in a natural way. This is because the face casts many small shadows that a human artist would not typically draw contours around.

VI. Acknowledgment

We would like to thank Prof. Amit Sethi of the Electrical Engineering Department, for giving me an opportunity to work on this project.

VII. References

[1] A Gentle Introduction to Bilateral Filtering and its Applications: https://people.csail.mit.edu/sparis/bf_course/

[2] Bilateral Filtering for Gray and Color Images: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

[3] Fast Bilateral Filtering: http://cs.brown.edu/courses/csci1290/2012/lectures/bf_course_Brown_Oct2012.pdf

[4] Bilateral Filters, Digital Visual Effects, Yung-Yu Chuang: https://www.csie.ntu.edu.tw/~cyy/courses/vfx/10spring/lectures/handouts/lec14_bilateral_4up.pdf

[5] Real-Time O(1) Bilateral Filtering: http://vision.ai.illinois.edu/publications/yang_cvpr09.pdf

[6] Toonify: Cartoon Photo Effect Application: https://stacks.stanford.edu/file/druid:yt916dh6570/Dade_Toonify.pdf

[7] Cartoonifying an Image, By Narendran Anil, http://inside.mines.edu/~whoff/courses/EENG510/projects/2013/Anil_slides.pdf

[8] Digital photography with flash/no-flash pair: http://hhoppe.com/proj/flash/

VIII. Code

The code can be found here

IX. Our Work in Core Point Detection

Our project started off with core point detection. We began by reviewing papers on core point detection. We started with “Fast and accurate algorithm for core point detection in fingerprint images” Bahgat et al., which we started implementing. The problem comes up when the smoothed orientation map needs to be segmented, and there seemed to be no segmentation method outlined in the paper. We then continued to go through “A Novel Fingerprint Singular Point Detection Algorithm” Bo et al. This paper had no specification on the value of ‘G’. On testing out with the same definition as given in the paper of Bahgat et al, it did not give us the desired result. “Core Point Detection using Improved Segmentation and Orientation” also had a similar problem. Moreover, before we switched papers, we looked for core point data. Sadly, we could not find any labelled dataset for corepoint. In order to circumvent this problem, we learned to annotate the dataset. We used multiple websites to no avail; dataturks does not work for points; LabelMe did not provide a convenient way to upload images to the set; another such website had a problem in displaying data uniformly. Finally, we ended up using a tool called LabelImg. However, this tool could only mark rectangular bounding boxes. We finally made small rectangles, around the order of pixel level, carefully annotating on all images of the FVC2002 dataset. These were saved in XML format, as few rectangles. We then learned XML parsing with python in order to extract the rectangle’s points, averaged them and “floored” them to integer values, in order to get a single point on all of the images. Unfortunately this data was rendered useless when we had to switch papers/