A comparison of image sharpness metrics and real-time sharpening methods with GPU implementations

Jason P. de Villiers[*]
Council for Scientific and Industrial Research
Meiring Naude Rd, Pretoria, South Africa

Abstract

Improving the quality of an image increases the probability, speed and accuracy with which possible objects of interest can be located and identified. Image sharpening, in particular, can correct for soft focus and strengthen the outlines of objects thus improving the identification and segmentation both by automatic means and by man in the loop systems. Output pixel independence is ensured so that a GPU can be used to sharpen the pixels in parallel, achieving processing performance increases of 20-360 fold. This work provides a metric which can quantify the sharpness of an image and shows that the sharpness of live video can easily be doubled in real-time on commercial desktop computers without inducing excessive noise.

CR Categories: I.4.0 [Image Processing and Computer Vision]: General—Image Processing Software; I.4.3 [Image Processing and Computer Vision]: Enhancement—Grey-scale Manipulation;

Keywords: Image Sharpness, Image Sharpening, GPU, real-time

1 Introduction

Sharpness is a measure of how in focus an image appears. Sharper images show more fine detail allowing for better object detection, recognition and classification [Bachoo and de Villiers 2009]. Image sharpening methods are broadly classifiable as either spatial or frequency domain algorithms, with the former being further broken down into first or second derivative based techniques [Gonzalez and Woods 2002]. This work focuses primarily on the frequency and first derivative spatial domain techniques. In the image spatial domain, sharpness is associated with the rate of change from one gray level to another [Gonzalez and Woods 2002], the fewer pixels this takes, the more distinct the edge appears. This also means it is easier to determine which pixels belong to which object, thus aiding segmentation. In the frequency domain, it is the higher frequencies that increase sharpness [Gonzalez and Woods 2002].

As with most problems in image processing, it is difficult to separate the sharpness of the image from the sharpness of a scene. If a scene has no distinct edges, and smooth blending between colour levels then the enhancement of it will not be sharp no matter the amount of image processing applied. Thus sharpness is best evaluated as a relative improvement over the original image, and not in trying to adjust the image to some fixed sharpness value.

With the advent of the increased progammability of Graphics Processing Units (GPU) and their seemingly ever increasing number of processor cores (the dual-GPU NVidia GTX295 has 480 cores), they are being increasingly applied to non-graphics applications. It is not uncommon to see processing performance increased 100 fold or more [Owens et al. 2008] when compared to a modern Central Processor Unit (CPU) implementation. There are still certain constraints as to what type of algorithms may be run on GPUs, [Owens et al. 2008] provide a thorough discussion of these. The most important is that each output pixel, while dependant on any number of (hopefully clustered) input pixels, is independent of other outputs. Table 1 provides the specifications of the GPU's used in this work. Low, mainstream and high-end are represented.

[*]e-mail: jdvilliers@csir.co.za

Table 1: *GPU Comparison*

	Num Shaders	GPU Clock (MHz)	Memory Bus Width (bits)	Memory Bus Speed (MHz)
Quadro MDS 140M	16	400	64	700
ATI HD 2400XT	40	800	64	700
NVidia 9600GT	64	650	256	900
NVidia GTX280	240	602	512	1107

2 Metric descriptions

Three metrics are used to evaluate images for sharpness. The first two are a measure of how much information is present in the image. Sharper images will use a greater palette than blurred images (which will eventually tend to the average intensity of the image as the blurring is increased). The third is a measure of the average edge strength, since images with strong edges visually appear more crisp (see Section 5).

2.1 First Order Entropy

In information theory, the Shannon Entropy (Eq (1)) provides a measure of the average information contained in the image, expressed as the number of bits (hence the base of 2 for the logarithm) required to represent each pixel - assuming that they're independent. The theoretical maximum entropy is the same as the bit depth of the image, which is 8 for the data used in this paper.

$$Entropy_1 = \sum_{i=0}^{255} -p(i) \times log_2\left(p(i)\right) \tag{1}$$

where:

$p(i) =$ the probability of a pixel having intensity i and is determined from the normalized 8-bit histogram.

2.2 Second Order Entropy

Eq. (2) provides a measure of the information when pairs of pixels in the image are considered. The number of possible pairs to use for calculating the histogram is $\sum_{i=1}^{n-1}[i]$ where n is the number of pixels in an image. Note that since pairs of pixels (assumed to be 8 bit grey-scale in this paper) are being considered there are

$2^{8+8} = 65536$ entries in the histogram.

$$Entropy_2 = \frac{1}{2}\sum_{i=0}^{65535} -p(i) \times log_2\left(p(i)\right) \tag{2}$$

where:

$p(i) =$ the probability of the 1st pixel having an intensity of $i >> 8$ and the 2nd having an intensity of $i\&0xFF$ as per the normalized 16-bit histogram.

Since pixels in different regions of the image are unlikely to be interdependent, a more realistic scenario in terms of processing requirements can be used. This is to calculate the entropy over each horizontally adjacent pair of pixels (there are $\approx n$ such pairs) and then repeat this for vertically adjacent pixel pairs. The geometric mean of these two values (as expressed by Eq. (3)) provides a measure of the information of each pixel given its four neighbouring pixels.

$$Entropy_{2adj} = \sqrt{Entropy_{2h} * Entropy_{2v}} \tag{3}$$

where:

$Entropy_{2h} =$ Eq. (2) considering only horizontally adjacent pixel pairs, and

$Entropy_{2v} =$ Eq. (2) considering only vertically adjacent pixel pairs.

2.3 Least Squares Error Surface Gradient Magnitude

The idea here is to approximate the local area in an image with a tractable function, such as the second order two dimensional polynomial of Eq. (4). Thereafter the partial derivatives with respect to the horizontal and vertical axes can be explicitly calculated, and the magnitude of the gradient vector determined, as shown by Eq. (5). This is similar to the method proposed by [Lucchese and Mira 2002] of fitting a second order surface in the vicinity of a checker intersection and solving for the saddle point to find is sub-pixel position.

$$f(x,y) = C_0 + C_1x + C_2y + C_3x^2 + C_4xy + C_5y^2 \tag{4}$$

where:

$f(x,y) =$ approximate intensity at (x, y) in the image.

$$|\bigtriangledown f(x,y)| = \sqrt{\left(\frac{\delta f(x,y)}{\delta x}\right)^2 + \left(\frac{\delta f(x,y)}{\delta y}\right)^2} \tag{5}$$

where:

$|\bigtriangledown f(x,y)| =$ magnitude of image intensity gradient at (x, y),

$\frac{\delta f(x,y)}{\delta x} = C_1 + 2C_3x + C_4y$, and

$\frac{\delta f(x,y)}{\delta y} = C_2 + 2C_5y + C_4x.$

In this work a 7 by 7 window centered on each pixel was used and therefore this is an over determined system of the type given by Eq. (6) where row-major multiplication is used. Unless the points in the window happen to lie on a second order two dimensional polynomial surface then there will be no solution for $\vec{C}$ which satisfies Eq. (6) perfectly, and so a best fit compromise for $\vec{C}$ must be found.

$$A\vec{C} = \vec{B} \tag{6}$$

where:

$$\vec{C} = \begin{bmatrix} \text{constant term} \\ x \text{ coefficient} \\ y \text{ coefficient} \\ x^2 \text{ coefficient} \\ xy \text{ coefficient} \\ y^2 \text{ coefficient} \end{bmatrix}$$

$A =$ matrix of pixel coordinate powers as per Eq (7), and

$\vec{B} =$ vector of pixel intensities per Eq (8),

The (49 by 6) matrix A, given by Eq. (7), is simply a list of the various combinations and powers of x and y as defined by Eq. (4) and in the same order as the definition of the column vector $\vec{C}$, calculated for every pixel in the window.

$$A = \begin{bmatrix} 1.0 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 \\ 1.0 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 \\ 1.0 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.0 & x_{i-1} & y_{i-1} & x_{i-1}^2 & x_{i-1}y_{i-1} & y_{i-1}^2 \end{bmatrix} \tag{7}$$

where:

$i =$ the number of pixels to fit the surface over, and

$(x_n, y_n) =$ the position of the nth pixel.

The column vector $\vec{B}$ (Eq. (8)) is a list of the intensities of the pixels who's coordinates were used to generate matrix A.

$$\vec{B} = \begin{bmatrix} I(x_0,y_0) \\ I(x_1,y_1) \\ I(x_2,y_2) \\ \vdots \\ I(x_{i-1},y_{i-1}) \end{bmatrix} \tag{8}$$

where:

$i =$ the number of pixels to fit the surface over,

$(x_n, y_n) =$ the position of the nth pixel, and

$I(x,y)) =$ the intensity of the pixel at (x,y).

In addition to simplifying the maths, fitting the surface of Eq. (4) in the least squares error sense will also smooth out any erroneous noise in the area. Eq. (9) shows how the coefficients can be determined analytically.

$$\vec{C} = \left(A^TA\right)^{-1} A^T\vec{B} \tag{9}$$

where:

$A =$ as defined in Eq. (7),

$\vec{B} =$ as defined in Eq. (8), and

$\vec{C} =$ as defined in Eq. (6).

It is more optimal to use relative coordinates (i.e. $x, y \in [-3, 3]$ for the 7 by 7 window used) in Eq. (7) instead of global image coordinates. This allows A, A^T, and $(A^TA)^{-1}$, to be calculated once. Additionally, the partial derivatives at the center pixel (i.e. $(0,0)$), simplify to C_1 for the x derivative and C_2 for the y derivative. Therefore the other four elements of $\vec{C}$ need not be calculated and so only 12 of the 36 multiplications required by $((A^TA)^{-1}) \times (A^TB)$ need be performed.

The final metric used (Eq. (10)) is the average of Eq. (5) over all pixels in the image whose 7 by 7 window lies entirely within the image.

$$AveGrad = \frac{1}{(w-6)\times(h-6)} \sum_{n=3}^{h-4}\sum_{m=3}^{w-4} |\bigtriangledown f(x_m, y_n)| \tag{10}$$

where:

$$\begin{aligned} |\bigtriangledown f(x_m, y_n)| &= \text{as defined in Eq. (5), and} \\ (w,h) &= \text{resolution of the image} \end{aligned}$$

3 Algorithm Descriptions

Image processing algorithms can be divided into two groups: spatial domain and frequency domain algorithms. Spatial domain algorithms operate directly on the pixels. Frequency domain algorithms are the extension to two dimensions of the algorithms used in signal processing theory. Low frequencies contain information about the general shape and colour of objects in the scene. Higher frequencies define the edges.

3.1 Spatial Domain Algorithms

For image sharpening in the spatial domain, the general idea is to increase the rate of change of pixel intensity in the vicinity of edges. Three algorithms are discussed below that do this.

3.1.1 Unsharp Masking

Details regarding unsharp masking (UM), originally developed in dark rooms to improve the contrast of photos captured on film, can be found in any good book on image processing (e.g. [Gonzalez and Woods 2002]). The image is compared to a slightly blurred version of itself. This blurred version by definition has both noise and edges made less distinct, whereas uniform areas of the image will be largely unaffected. The difference between the original image and the blurred image is then added back to the original image thus emphasizing the edges. In the dark room this was done by developing the original film stacked on top of the negative of the blurred version. Unsharp masking is expressed mathematically as:

$$\begin{aligned} I'(x,y) = I(x,y) + G\big(I(x,y) - \\ \frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n} I(x+i, y+j)\big) \end{aligned} \tag{11}$$

$$\begin{aligned} &where: \\ I'(x,y) &= \text{the new intensity at } (x,y) \text{ in the sharpened image} \\ I(x,y) &= \text{the intensity of the input image at } (x,y) \\ G &= \text{the specified, constant gain used to sharpen the image} \\ n &= \text{the window size used for the sharpening} \end{aligned}$$

This algorithm does not actively look for edges and enhance only in their vicinity. Instead, pixels near the edge of a uniformly shaded region are compared to an average which includes pixels that are not part of the region, and thus its intensity will be modified more than pixels in a uniform region. This algorithm also causes noise to be amplified in regions that should be uniform but are not. The selected gain is then a compromise between enhancing the edges and making the image look grainy. Small window sizes highlight fine detail, whereas large window sizes enhance the main regions, but may suppress detail near the edges of region boundaries.

3.1.2 Standard Deviation Gain

Standard Deviation Gain (SDG) is similar to the previous algorithm, except its gain is not constant. It aims to have a high gain in the vicinity of a region boundary and a low gain in uniform areas of the image, as shown in Eq (12):

$$\begin{aligned} I'(x,y) = I(x,y) + f(x,y)\big(I(x,y) - \\ \frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n} I(x+i, y+j)\big) \end{aligned} \tag{12}$$

$$\begin{aligned} &where: \\ I(x,y) &= \text{the intensity of the input image at } (x,y) \\ I'(x,y) &= \text{the new intensity at } (x,y) \text{ in the sharpened image} \\ f(x,y) &= \text{variable gain dependant on the local image content} \\ n &= \text{the window size used for the sharpening} \end{aligned}$$

The particular gain considered here is a damped version of the standard deviation (Eq. (13)). This is similar to the method of Kwon and Liang [Kwong and Liang 1992], although they use the variance for unsharp masking and not as a differential gain. In constant regions the standard deviation will be low, near edges it will be high. However it can be extremely high near edges and hence the damping, which was empirically determined and may need to be scaled if the intensities are not represented by 8 bit unsigned integers.

$$f(x,y) = ln\left(\sqrt{g(x,y) - (h(x,y))^2}\right) \tag{13}$$

$$\begin{aligned} &where: \\ g(x,y) &= \frac{1}{(2n+1)^2}\sum_{i=y-n}^{y+n}\sum_{j=x-n}^{x+n} (I(i,j))^2 \\ h(x,y) &= \frac{1}{(2n+1)^2}\sum_{i=y-n}^{y+n}\sum_{j=x-n}^{x+n} (I(i,j)) \\ f(x,y) &= \text{variable gain dependant on the local image content} \\ I(x,y) &= \text{the intensity of the input image at } (x,y) \\ n &= \text{the window size used for the sharpening} \end{aligned}$$

3.1.3 Sobel Gain

This algorithm is also based on Eq. (12), however it uses the magnitude of the local intensity gradient vector (as estimated using the Sobel operator [Sobel 1970]) as its variable gain function. This value too is scaled, so as to prevent excessive gain from binarizing the image near edges. This damping was empirically determined and may need to be scaled if the intensities are not represented by 8 bit unsigned integers.

$$f(x,y) = 1 + ln\left(\sqrt{\left(\frac{\delta}{\delta x}I(x,y)\right)^2 + \left(\frac{\delta}{\delta y}I(x,y)\right)^2}\right) \tag{14}$$

$$\begin{aligned} &where: \\ I(x,y) &= \text{the intensity of the input image at } (x,y) \\ f(x,y) &= \text{variable gain dependant on the local image content} \\ n &= \text{the window size used for the sharpening} \end{aligned}$$

3.2 Frequency Domain Algorithms

Frequency domain algorithms work by manipulating the power spectral density of the image, the phase is left unchanged. In magnitude and phase representation, the distance of a point in the spectral domain from the centre of the image corresponds to its frequency, and its polar angle corresponds to the angle of the sine wave in the image.

3.2.1 Converting To and From the Frequency Domain

To convert from the time domain to the frequency domain, the Fourier transform is used. In 1807 Jean Baptiste Joseph Fourier showed that any repetitive signal could be represented as an infinite series of harmonic sine waves. The specific magnitudes and phases of the sine waves are what made each signal unique.

The details of the Discrete Fourier Transform (DFT) and its more efficient counterpart, the Fast Fourier Transform (FFT), can be found in any handbook on signal processing [Carlson 1998] or image processing [Gonzalez and Woods 2002]. Suffice it to say that the FFT requires $O(Nlog_2N)$ operations whereas the DFT requires $O(N^2)$ operations. This results in a speed up from less than 0.03 Frames Per Second (FPS) to 2.75FPS when converting a 640 by 480 pixel image on the CPU. The GPU implementation of the FFT [Moreland and Angel 2003] provided a further increase from 2.75FPS to 57FPS.

3.2.2 Mid Frequency Boost

In an image, the zero frequency is the average intensity, the highest frequencies tend to be noise, and the low frequencies contain the basic shape information. The fine details and crisp edge information is contained in the high frequencies. So this algorithm merely multiplies all the frequencies in a chosen band by a specified gain, and hence its name the Mid-Frequency Boost (MFB). As is always the case in the frequency domain, the band edges must be smooth and continuous so as to minimize ringing and echo ripple effects in the image. The band used in this evaluation is the 0.2 to 0.8 band (normalized to maximum image frequency). The bandpass filter was created from 6th order low and high pass Butterworth filters with-3dB cut off frequencies of 0.8 and 0.2 respectively

4 Results

This section provides all the results of the comparisons of the sharpening algorithms considered in this study. Two 640 by 480 8-bit grey-scale images were used, one with relatively little detail/edges (Figure 1(a) and 3(a)) and one with a large amount of detail/edges (Figure 2(a) and 3(b)).The sharpened images, the quantified improvement in sharpness, and comparative processing rates are provided. Table 2 indicates, for relative comparison purposes, the average amount of time taken for 100 iterations of each metric on an Intel Atom N270 1.6GHz CPU. Table 3 summarizes the results of the different sharpening algorithms. The results are as per the output of Eqs (1), (2) and (10), the first two have units of bits per pixel, and the last shades per pixel.

Table 2: *Metric execution time*

Metric	Execution Time (ms)
$Entropy_1$	4.58
$Entropy_{2adj}$	18.55
Intensity Gradient	5780605.0

Table 4 provides the comparative rates of the different algorithms on the different hardware platforms, as well as the speed up factor of the NVidia GTX280 over the CPU. The CPU used was a (single core of) an Intel Core2 Quad 2.83GHz, the system had 4GB of RAM although the 32 bit operating system only allowed access to 3.2GB. OpenSceneGraph was used to handle the loading of the GPU programs (shaders), and its built in statistics collector was used to obtain the rates. The shaders were written in the OpenGL Shading Language allowing them to run on both NVidia and ATI GPUs. The rates quoted for MFB include conversion to and from the frequency domain, which may already be part of the specific image processing chain.

Figures 1 and 2 show the results for the different spatial domain sharpening techniques, omitting window size $n = 7$ for the unsharp mask and SDG algorithms due to space constraints. Figure 3 shows the results of two different gains for the MFB algorithm for both the high and low detail image.

5 Discussion of results

A visual inspection of Figures 1, 2 and 3 indicates that MFB and SDG induce the least amount of noise in the image, while still enhancing the images. This correlates well with both entropy metrics for both the high and low detail images in Table 3. Similarly, the image gradient metric corresponds well to the edge strength as expected, but does not indicate any degradation in the image. It is seen from Table 2 that the two entropy metrics are much quicker gradient intensity metric. It can be concluded that the average image gradient is not a suitable metric for sharpness, regardless of the method used to generate it (e.g. sum of the Sobel gradient intensities), since few methods are as accurate as that in 2.3.

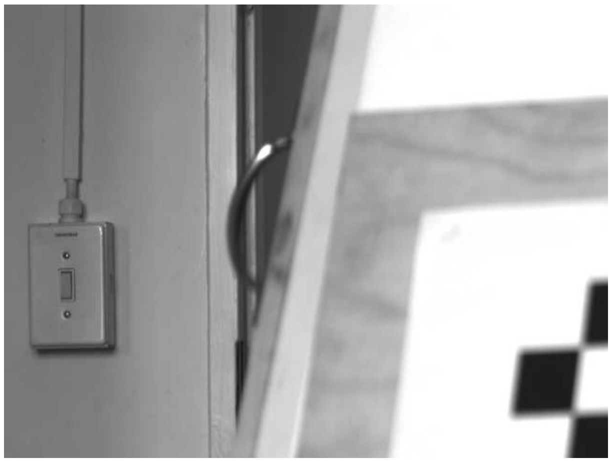
The range of results for $Entropy_1$ is the lowest having a -0% to +4% range for the low detail image, compared to -0% to +12% and -0% to +125% for the $Entropy_{2adj}$ and intensity gradient metrics respectively. For the high detail image these ranges are -23% to +1%, -21% to +7% and -0% to +129% for the $Entropy_1$, $Entropy_{2adj}$ and intensity gradients respectively. It is perhaps surprising that the two entropy metrics rate algorithms in very similar orders if they are ranked by entropy, especially since the adjacency inherent in $entropy_{2adj}$ intuitively feels like it should be more sensitive to edges and thus perceived sharpness.

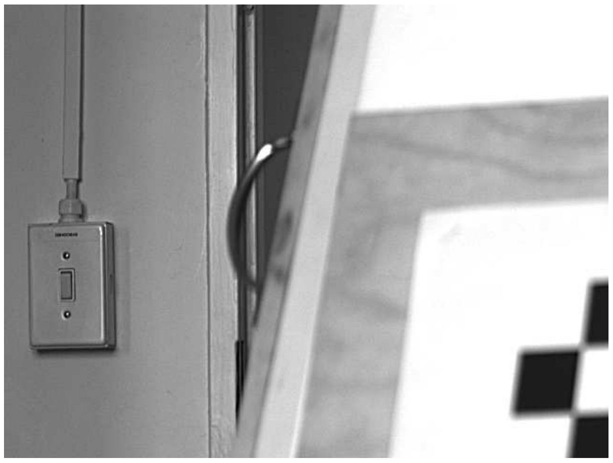
As can be seen from Figures 1(b), 1(c), 2(b), and 2(c) increased window sizes causes unsharp masking to make the edges more distinct, however image detail in the vicinity of the edges is lost. Additionally, induced graininess is visible in the smooth regions of the low detail image. This is reflected in Table 3 where the Intensity Gradient increases with window size, yet the overall entropy decreases, indicating a loss of information.

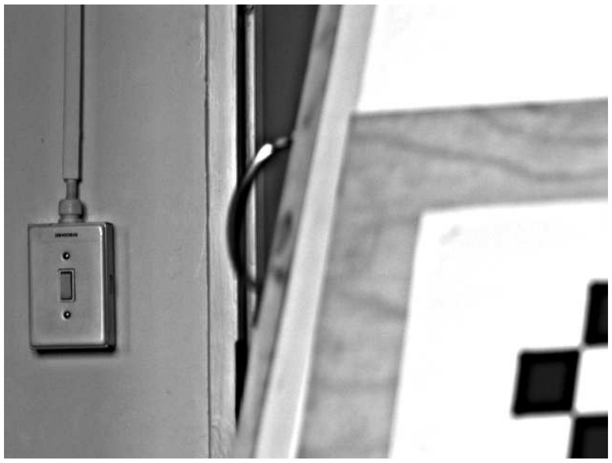
Figures 1(d), 1(e), 2(d), and 2(e) show differing results for the low and high detail images using the SDG algorithm. For the low detail image, sharpening it with a window size of 3, produces a very crisp image, a fact not reflected by any of the three metrics. In addition it causes an unwanted halo-ing around the edges with large window sizes. This last is true for the high detail image too, although even the small window sizes provide too much amplification. However, in the constant intensity regions of the images it does not magnify the minor differences and cause additional graininess, and has superior performance to unsharp masking in this regard.

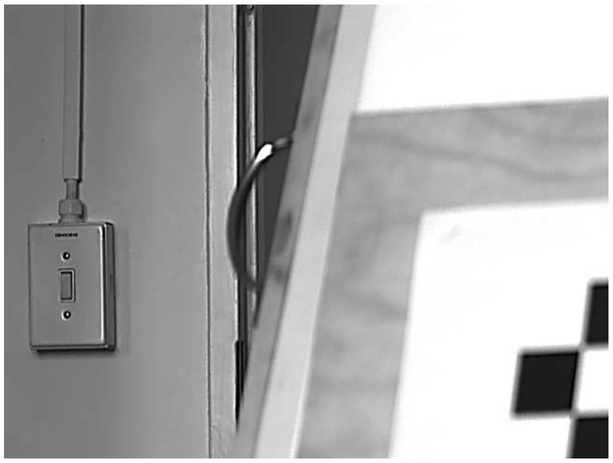
Figures 1(f) and 2(f) show the results for the Sobel-gain algorithm. This algorithm was consistently the quickest algorithm and provided the second highest improvements in entropy, with comparatively little degradation compared to the large window size unsharp masking and SDG algorithms.

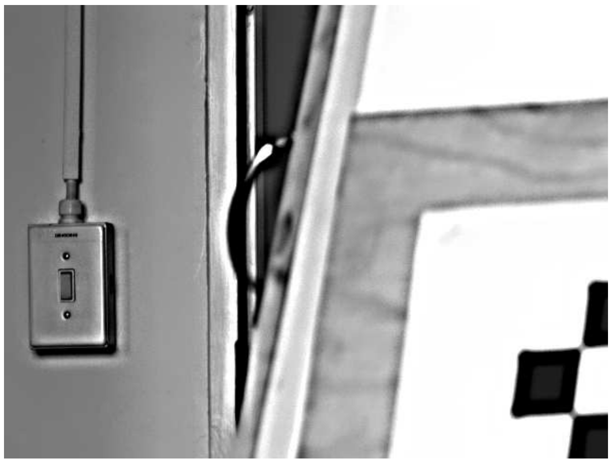
Subjectively, MFB (Fig 3) adds very little noise to the images, yet provides the highest increase in entropy (see Table 3), implying that the gain could be further increased safely. However, these al-


(a) Original Image


(b) Fixed, win size 3, gain 2


(c) Fixed, win size 15, gain 2


(d) StDev, win size 3, gain 2


(e) StDev, win size 15, gain 2

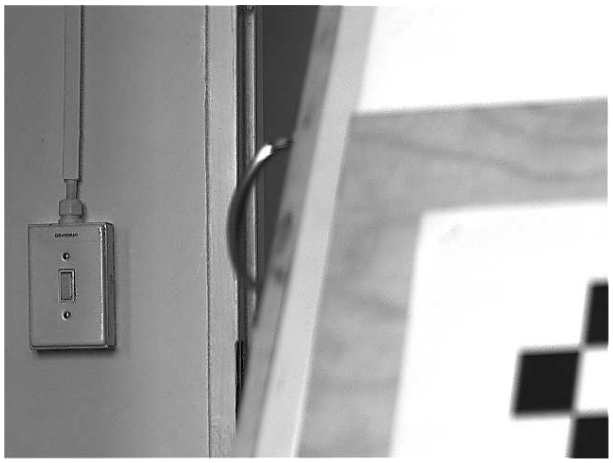
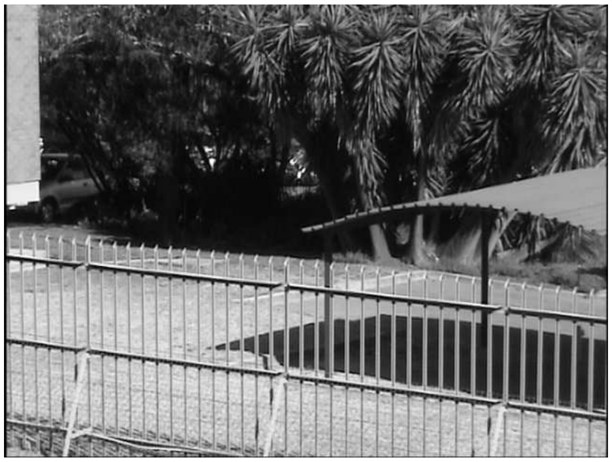
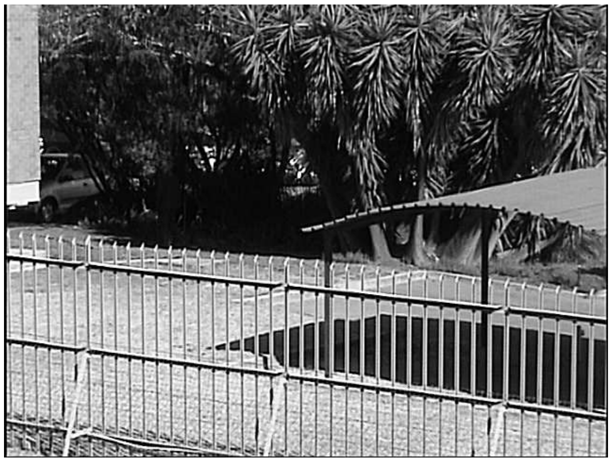
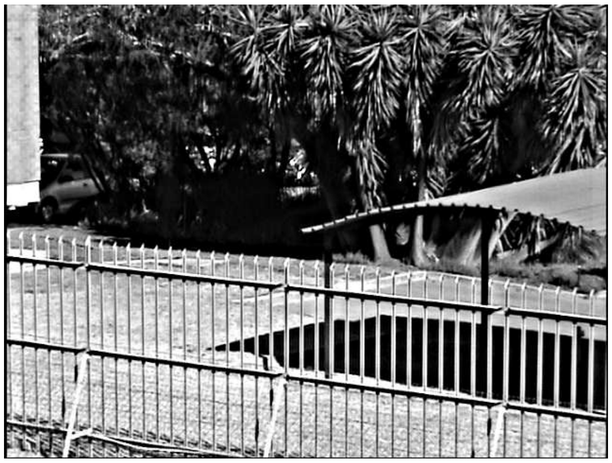
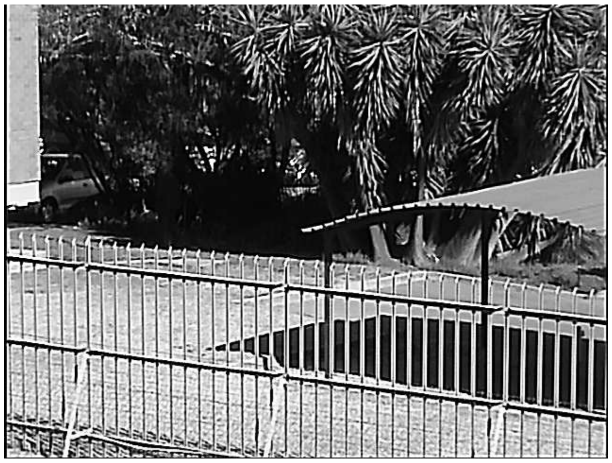

(f) Sobel Gain

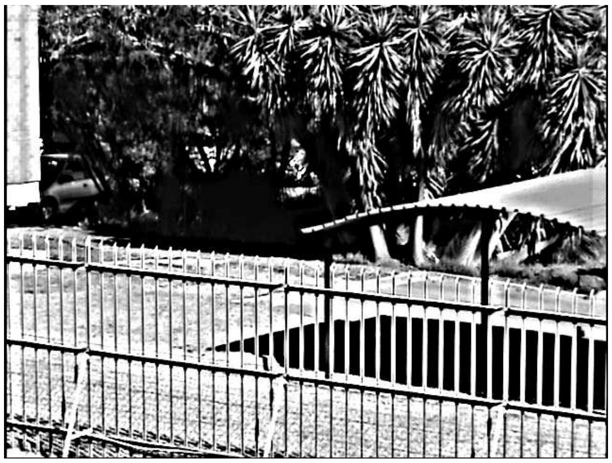
Figure 1: *Low detail image, spatial sharpenings*


(a) Original Image


(b) Fixed, win size 3, gain 2


(c) Fixed, win size 15, gain 2


(d) StDev, win size 3, gain 2


(e) StDev, win size 15, gain 2

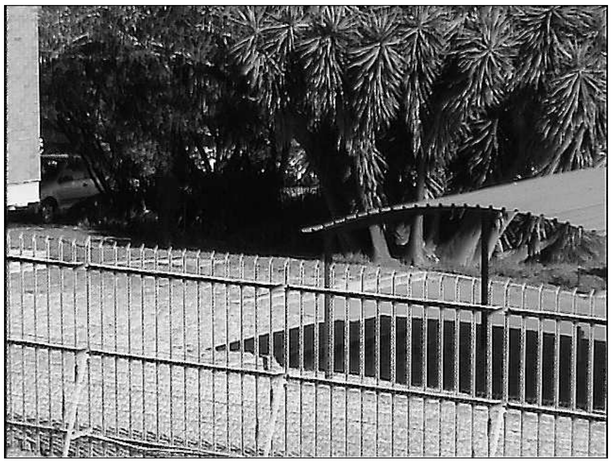
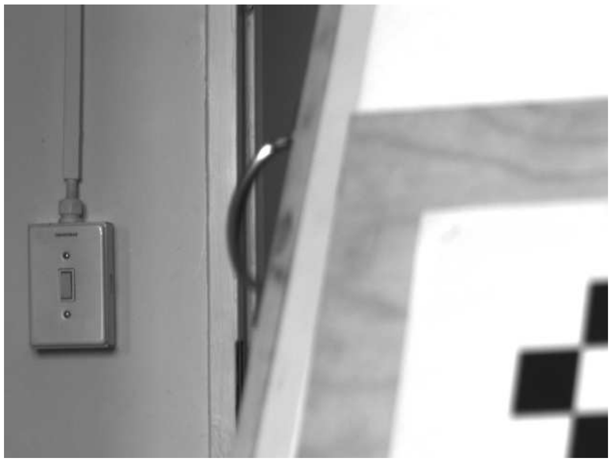
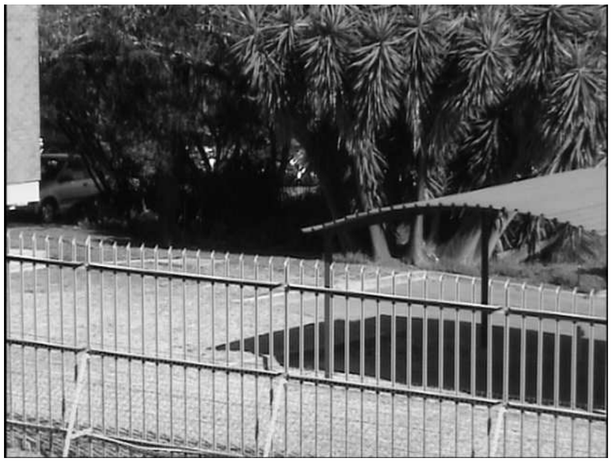
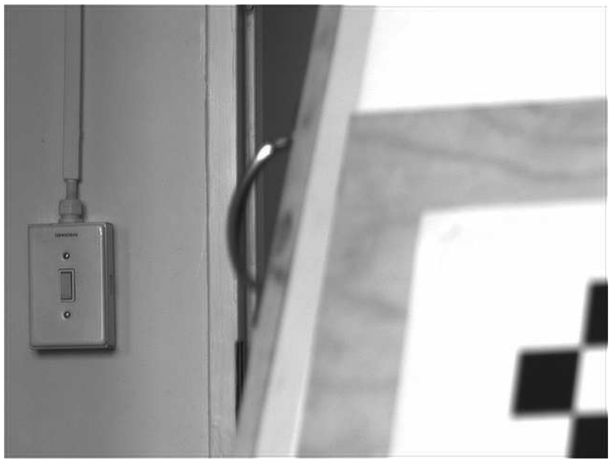
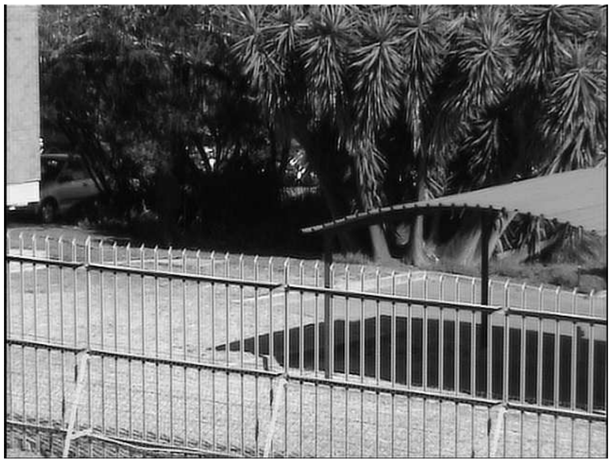

(f) Sobel Gain

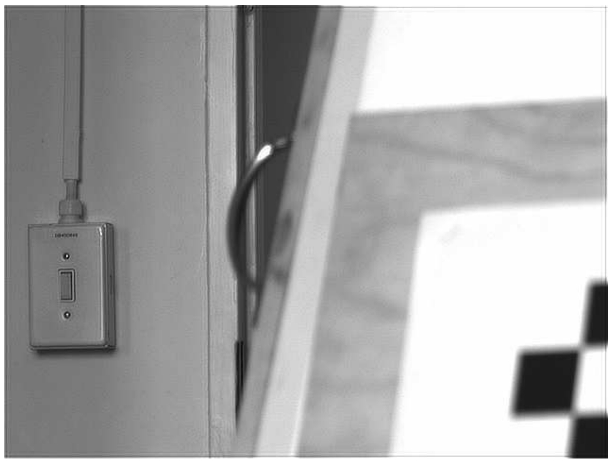
Figure 2: *High detail image, spatial sharpenings*


(a) Original Image


(b) Original Image


(c) MFB, gain 2


(d) MFB, gain 2


(e) MFB, gain 3

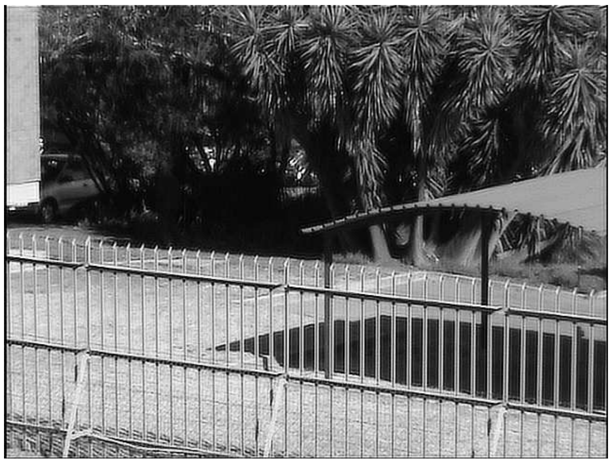

(f) MFB, gain 3

Figure 3: *Frequency domain sharpenings*

Table 3: *Sharpness metrics*

Algorithm Info			Low Detail Image			High Detail Image		
Name	Win Size	Gain	$Entropy_1$ (bits/pixel)	$Entropy_{2adj}$ (bits/pixel)	Intensity Gradient (shades/pixel)	$Entropy_1$ (bits/pixel)	$Entropy_{2adj}$ (bits/pixel)	Intensity Gradient (shades/pixel)
Orig			6.54	4.74	1.74	7.64	6.47	6.73
MFB		1	6.78	5.17	1.78	7.68	6.74	6.86
MFB		2	6.81	5.34	1.83	7.72	6.92	7.02
Sobel			6.65	5.20	1.95	7.48	6.75	8.53
UM	3	2	6.58	5.04	2.05	7.62	6.77	9.51
UM	7	2	6.59	5.04	2.58	7.43	6.55	11.79
UM	15	2	6.54	4.97	2.98	7.16	6.27	11.97
SDG	3		6.56	4.87	2.21	7.12	6.30	11.88
SDG	7		6.67	4.94	3.24	6.44	5.63	15.34
SDG	15		6.52	4.94	3.91	5.85	5.10	15.40

Table 4: *CPU vs GPU Processing Rate Comparison*

Model Name	Window Size	Gain	CPU (FPS)	Quadro MDS 140M (FPS)	ATI HD 2400XT (FPS)	NVidia 9600GT (FPS)	NVidia GTX280 (FPS)	GPU Improvement (factor)
Sobel			25.6	69.2	309.6	730.0	3100	121.1
MFB			2.75	3.75		33.6	57	20.7
UM	3	2	9.4	44.9	70.4	434.0	1140	121.3
UM	7	2	2.5	17.1	25.6	142.9	537	214.8
UM	15	2	0.6	2.6	6.3	24.5	154	256.7
SDG	3		5.5	43.8	70.2	450.1	1130	205.5
SDG	7		1.6	14.5	20.6	141.7	559	349.4
SDG	15		0.4	2.33	6.2	21.4	147	245.0

gorithms are comparatively slow, if the transformation to and from the frequency domain is considered.

Porting the algorithms to the GPU, considerably increased the performance of the algorithms with gains proportional to the specifications of the GPU's provided in 1. The two best sharpening algorithms, SDG and MFB, achieved processing gains of 120 and 20 fold respectively, with the former running at several thousand FPS.

6 Conclusions

The $entropy_1$ metric is both the quickest to calculate (especially if the histogram is already calculated for use elsewhere in the image processing pipeline), accurately reflects the improvement in image quality due to sharpening, and indicates when too much sharpening is being applied. The Sobel gradient based variable gain algorithm is the optimal choice in the spatial domain, providing results only slightly inferior to the mid-frequency boost. If the specific application already does processing in the frequency domain then a high-gain mid-frequency boost would be the optimal choice. High frame rates are easily achieved using even low-end GPUs, which can comfortably sharpen a video frame before the next arrives.

A suitable metric to measure image quality in a real-time manner has been found, and was used to evaluate several different sharpening algorithms. Furthermore, these algorithms have been ported to the GPU, yielding considerably quicker frame rates such that real-time processing of video is possible on even fairly low-end modern personal computers.

Acknowledgements

The author would like to thank the Armaments Corporation of South Africa (Armscor) for funding this research.

References

BACHOO, A., AND DE VILLIERS, J. 2009. Contrast enhancement. Tech. Rep. 6700-OPTO-38602-01, CSIR. Restricted document.

BUTTERWORTH, S. 1930. On the theory of filter amplifiers. *Wireless Engineer 7*, 635–541.

CARLSON, G. 1998. *Signal and Linear System Analysis*. John Wiley and Sons, Inc.

GONZALEZ, R., AND WOODS, R. 2002. *Digital image processing*. Addison-Wesley Publishing Company.

KWONG, H., AND LIANG, J., 1992. Unsharp masking using centre weighted local variance for image sharpending and noise suppression. US Patent num 5081692.

LUCCHESE, L., AND MIRA, S. 2002. Using saddle points for subpixel feature detection in camera calibration targets. In *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, vol. 2, 191–195.

MORELAND, K., AND ANGEL, E. 2003. The FFT on a GPU. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, 112–119.

OWENS, J., HOUSTON, M., LUEBKE, D., GREEN, S., STONE, J., AND PHILIPS, J. 2008. Gpu computing. *Proceedings of the IEEE 96*, 5, 879–899.

SOBEL, I. 1970. *Camera Models and Machine Perception*. PhD thesis, Stanford University, Palo Alto, California.