Chromatic Encoding: a Low Power Encoding Technique for Digital Visual Interface

Wei-Chung Cheng and Massoud Pedram
University of Southern California
Los Angeles, CA 90089
{wccheng,massoud}@sahand.usc.edu

Abstract

This paper presents a low-power encoding technique, called chromatic encoding, for the Digital Visual Interface standard (DVI), a digital serial video interface. Chromatic encoding reduces power consumption by minimizing the transition counts on the DVI. This technique relies on the notion of tonal locality, i.e., the observation that the signal differences between adjacent pixels in images follow a Gaussian distribution. Based on this observation, an optimal code assignment is performed to minimize the transition counts. Furthermore, the three color channels of the DVI may be reciprocally encoded to achieve even more power saving. The idea is that given the signal values from the three color channels, one or two of these channels are encoded by reciprocal differences with a number of redundant bits used to indicate the selection. The proposed technique requires only three redundant bits for each 24-bit pixel. Experimental results show up to a 75% transition reduction.

1 Introduction

In mobile computing, power is undoubtedly one of the most important design challenges. To address this issue, fruitful low power techniques have been developed. Memory bus encoding for low power is not only widely-studied in the academia [1-4] but also well-utilized by high performance microprocessors [5]. It minimizes transition counts on the off-chip memory buses to reduce the power dissipation. Cost-effective as it is, bus encoding has not been adopted by the peripherals such as LCD displays. The reason is that the power consumed by the peripheral devices eclipses that consumed by the peripheral interfaces, as the latter cannot be reduced by memory bus encoding. Nevertheless, the evolving of display technology changes the paradigm. For example, the self-emitting organic LED (OLED) or organic electroluminescence (EL) display provides wider viewing angle, higher brightness, and lower supply voltage, while consumes much less power than the conventional backlighted LCD [6]. EL-equipped electronics, such as cell phones, car audios, and digital cameras, are currently available. More advanced display devices are emerging and will take up a smaller portion of the entire power budget.

New standards for the next generation display devices are needed to replace the legacy analog VGA standards [7]. The Digital Visual Interface (DVI) is advocated by the Digital Display Working Group to provide a display-technology independent, high speed, digital interface [8]. A DVI 1.0 connection consists of one or two Transition Minimized Differential Signaling (TMDS) serial links. The voltage swings of the TMDS differential pairs are 0~780mV and –780~0mV. The maximum frequency is 1.65Gbps. The length of the DVI cable is limited to 4.6m. The typical mutual capacitance of high quality DVI cables is around 60pF/m [9]. Plugging these numbers in the approximate power equation $P=0.5CV^2f$, the transceiver of each DVI channel consumes 277mW, in the extreme case, to switch the cable capacitance alone. Typical off-the-shelf standard memory chips, e.g. Synchronous DRAM, work at 3.3V, 400MHz with around 15pF capacitance on each I/O pin. In other words, low power encoding techniques are more promising for the DVI than memory bus.

2 Digital Visual Interface

Consider a typical portable embedded system [10] as shown in Figure 1. The system consists of a mainboard, a display, and a DVI connection in between. The pixel data is stored in the frame-buffer by the CPU. The graphics controller fetches the pixel data from the frame-buffer and then generates the proper video signals through the DVI. The pixels on the display are scanned in a left-to-right, top-down fashion. Therefore, the data of two adjacent pixels on the same row will be sent through the channel consecutively.

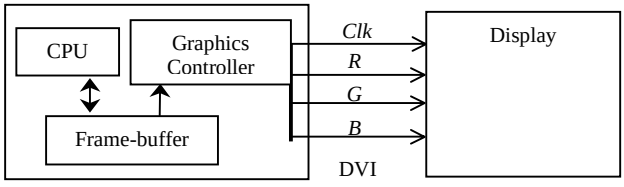


Figure 1: The target system.

The DVI consists of one or two *transition minimized differential signaling* (TMDS) links [8]. A TMDS link consists of one clock channel and three data channels for the red, green and blue (R, G, and B) components. Each pixel has three-color components in the range of 0 to 255. The TMDS transmitter encodes a 8-bit source-word $D[7..0]$ into a 10-bit code-word $E[9..0]$. The TMDS algorithm is meant to minimize the transition counts and works as follows. $E[0]$ is the same as $D[0]$. $E[8>i>0]$ is encoded as XOR($D[i]$,$E[i-1]$) or XNOR($D[i]$,$E[i-1]$). The decision between XOR and XNOR is

This research was supported in part by DARPA PAC/C program under contract DAAB07-02-C-P302 and by NSF under grant no. 9988441.

1530-1591/03 $17.00 © 2003 IEEE

specified by $E[8]$. Finally, the 9 bits of $E[8..0]$ can be selectively inverted, which is indicated by $E[9]$. The TMDS encoding algorithm can be considered the serial version of the transition signaling encoding algorithm [2], which does not reduce the transition counts if the data source is purely random.

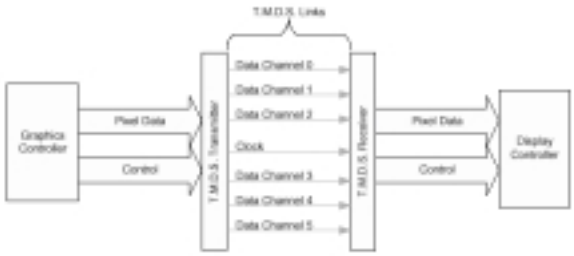


Figure 2: Digital Visual Interface [8].

Consider the transition counts on a single channel of the DVI. For each pixel i in the frame-buffer, its 8-bit signal value x_i is encoded as the 10-bit $e(x_i)$ and sent serially through the channel. The sum of the transition counts on the channel is $\sum_i \left(\sum_{j=0}^{8} e(x_i)[j] \oplus e(x_i)[j+1] + e(x_i)[9] \oplus e(x_{i+1})[0] \right)$. The first term represents the intra-word transitions, and the second term represents the inter-word transitions. Define w as the intra-word transition counts $w(x) = \sum_{j=0}^{8} x[j] \oplus x[j+1]$. The total transition counts can be re-written as $\sum_i \left(w(e(x_i)) + e(x_i)[9] \oplus e(x_{i+1})[0] \right)$. Since the inter-word transition counts occupy only a small portion of the total transition counts, this part is ignored during the derivation of the chromatic encoding algorithm. The goal is to find the optimal code assignment e to minimize $\sum_i w(e(x_i))$.

3 Tonal locality

In this section, the tonal locality claim is made -- a pixel is likely to be similar to its neighboring pixels in terms of the signal values.

3.1 Intuition

The tonal locality can be intuitively explained from the aspect of the edge detection techniques of digital image processing [11]. Given the set of the pixels on an image as a two-dimensional array of signals and consider its frequency space. Edge detection algorithms act as high-pass filters. The signals with high frequencies, i.e., the pixels located in the regions that present fast transition in tonality, pass through the filter and are reported as edges. The remaining pixels, such as those from surfaces of objects, have low frequencies and therefore have similar tone as their neighboring pixels.

Another source of tonal locality is image compression. The kernel of the image compression algorithms is the *inverse discrete cosine transform* (IDCT). The IDCT is applied to the luminance and chromaticity of each macroblock. After the transformation, the high-frequency coefficients, which represent the very fine details on the image that are hardly perceivable by human eyes, are discarded to reduce the information entropy. Therefore, the variation between adjacent pixels in the same macroblock is limited, though fast transitions may still exist between adjacent macroblocks.

3.2 Benchmark image suite

To justify the claim of tonal locality, the statistics were examined on a set of benchmark images from the USC SIPI Image Database (USID) [12]. The USID is considered the ad hoc benchmark suite in the signal and image processing research field [11]. It consists of three volumes (texture patterns, aerial photos, and miscellaneous) of color and black-and-white images in different sizes. The results reported here are from 8 color images from volume 3. All of them are 256 by 256 pixels. The color depth is 24 bits, i.e., 8 bits per color-channel in the range of 0 to 255. These images are uncompressed and stored in the TIFF (Tag-based Image File Format) format.

Correlation analysis was performed on these images. First, each image was decomposed into 3 channels. The coefficients of correlation between every pair of pixels within a 5x5 region were calculated. The coefficients of determination (the square of the coefficient of correlation) were then calculated. Table 1 shows the results of the benchmark image 4.1.01. The value in the table at row v and column u represents the coefficient between the pixel at location (i,j) and the pixel at location $(i+u,j+v)$.

Table 1: Coefficients of determination of 4.0.1.

R	*i*	*i+1*	*i+2*	*i+3*	*i+4*
j	1.00	0.97	0.93	0.88	0.84
j+1	0.96	0.95	0.91	0.87	0.83
j+2	0.90	0.90	0.87	0.84	0.81
j+3	0.85	0.84	0.83	0.81	0.78
j+4	0.79	0.79	0.79	0.77	0.75

G	*i*	*i+1*	*i+2*	*I+3*	*i+4*
j	1.00	0.97	0.92	0.89	0.85
j+1	0.96	0.95	0.91	0.88	0.85
j+2	0.92	0.91	0.89	0.86	0.83
j+3	0.88	0.87	0.86	0.84	0.82
j+4	0.84	0.84	0.83	0.81	0.79

B	*i*	*i+1*	*i+2*	*i+3*	*i+4*
j	1.00	0.95	0.91	0.87	0.83
j+1	0.95	0.93	0.90	0.86	0.83
j+2	0.91	0.90	0.87	0.85	0.82
j+3	0.87	0.86	0.85	0.82	0.80
j+4	0.84	0.83	0.82	0.80	0.78

According to the above data, a pixel is highly correlated (≥0.95) to its adjacent pixels in each channel. The coefficients decrease gradually as the distance increases.

To exploit the above correlation, the approach presented here is to use the neighboring pixels to predict the present pixel. The remaining errors will be coded and sent over the channels. The more accurate the predictor is, the fewer transition counts the code-words will have. The predictor gains more accuracy if more reference pixels are used at the cost of memorizing the reference pixels. However, while scanning an image, only half of the 8 neighboring pixels have been processed by the encoder (northwest, north, northeast, and west). How many pixels are sufficient to build an accurate predictor? To answer this question, 1-, 2-, 3-, and 4-way multiple linear regression analysis was performed on the 8 benchmark images. The independent variables were chosen from the four neighboring pixels (west, north, northwest, and northeast) incrementally. Table 2 shows the average of the standard deviation of the residues of the three color channels under confidence level of 0.95.

Table 2: Multiple regression analysis.

	1-way	2-way	3-way	4-way
4.1.01	11.0130	8.7331	8.2357	8.0786
4.1.02	12.0754	8.2141	7.0998	7.0313
4.1.03	6.8144	6.2441	5.3144	5.2100
4.1.04	15.8660	8.4618	5.8766	5.8178
4.1.05	11.0146	8.6054	6.3802	6.2936
4.1.06	16.6531	13.5351	11.9507	11.7988
4.1.07	8.0694	5.8347	4.4617	4.3720
4.1.08	10.3856	7.7344	5.6498	5.4513

The above data show that while the two-way predictor outperforms the one-way predictor, three and four-way predictors do not gain much accuracy. However, to implement the two-way predictor, all of the pixels on the previous (north) row, usually hundreds, have to be stored. Due to the extra buffers and power overhead required, a one-way predictor, i.e., encoding against the previous (west) pixel, will be used here.

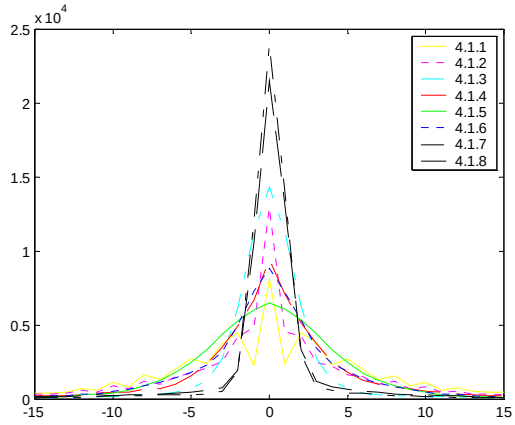


Figure 3: The distributions of the signal value differences.

Figure 3 shows the distributions of the signal value differences, $x_i'=x_i-x_{i-1}$, of the benchmark images. In fact, three curves from each color channel of each image are plotted. However, they are completely overlapped and only 8 curves are shown. The curves follow Gaussian distributions with the same means equal to zero and different standard deviations. Most of the differences are within the range of [-15,15].

3.3 Code assignment

If the statistics of the pixel signals are known, then the total transition count can be written as $T=\sum_{x=0}^{255} p(x)w(e(x))$, where p is the occurrence of signal x. The objective is to find an optimal assignment e such that T is minimized. Based on the tonal locality, the differences between consecutive pixels follow a Gaussian distruibution. If the difference encoding is used, the total transition count can be written in terms of the signal differences x': $T=\sum_{x=-255}^{255} p(x')w(x')\cdot p(x')$, which follows a Gaussian distribution, has the property $p(x)\le p(y)$ if $|x|>|y|$. The optimal code assignment will be any function f that satisfies $f(x)\le f(y)$ if $|x|<|y|$. Functions in this class are called *ordered transition codes* (OTC). If the length of the code-words is N, then there are $2\binom{L-1}{i}=\frac{2(L-1)!}{i!(L-1-i)!}$ code-words having exactly i intra-word transitions.

The code assignment is generated as a two-column lookup table. In the first column, all of the source-words are enumerated and sorted by their magnitudes. In the second column, all of the code-words (E[7..0]) are enumerated and sorted by their intra-transition counts. The mapping from the first column to the second is the encoding function. The reverse mapping is the decoding function.

4 Encoding framework

The presented encoding framework consists of three steps: spatial encoding, chromatic encoding, and codebook lookup. The spatial encoding subtracts the previous (west) pixel value from the current one, the chromatic encoding takes the differences between the three color channels, and the codebook lookup encodes the signed binary numbers into the OTC.

4.1 Spatial encoding

The spatial encoding predicts the current pixel value x_i based on the pixel in the west x_{i-1}. According to the scanning order, x_{i-1} and x_i are two adjacent pixels on the same row except for the boundary conditions. The prediction error $x_i'=x_i-x_{i-1}$ is calculated as the output and its range is extended to [-255,255]. To store the previous pixel value, only one buffer is needed for each color channel.

4.2 Chromatic encoding

The three values from each channel can be encoded by their reciprocal differences to reduce the transition counts. Recall the monotonously increasing property of the OTC: $f(x)\le f(y)$ if $|x|<|y|$. To minimize the transition counts of the code-words, our encoding strategy is to minimize the magnitudes of the source-words. The reciprocal differences are calculated and compared against the original values. The three out of six values with the smallest magnitudes will be chosen as the code-words and sent along with the redundant bits.

The selection problem can be formulated as an equivalent graph problem. Let *r*, *g*, and *b* be the three values from the three color channels. Consider a weighted clique of four vertices {*a*, *b*, *c*, *o*}. The weights of *oa*, *ob*, and *oc* (called *inputs*) are *r*, *g*, and *b*, respectively. The weights of *ab*, *bc*, and *ca* (called *differences*) are the differences *g*–*r*, *b*–*g*, and *r*–*b*, respectively. Among these six edges, in order to represent the original values *r*, *g*, and *b*, exactly three edges are needed, and they must form a tree. The optimal solution is the tree with the minimum weight, i.e., the minimum spanning tree [13].

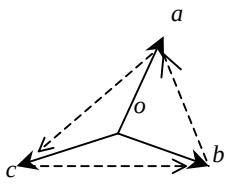


Figure 4: The minimum spanning tree problem.

Since {*a*, *b*, *c*, *o*} are actually scalars, they can be sorted into 24 cases as shown in the first column of Table 3. The optimal selection is simple to determine. The 24 cases can be reduced to 12 cases because each case has a dual, which is in reverse order and has the same optimal selection. These 12 cases can again be categorized into two types. In the first type ($0<x<y<z$ or $z>y>x>0$), including cases 1~6, one input (*x*) and two differences (*y*-*x* and *y*-*z*) are chosen. *x* is indicated by the redundant bits. The two differences are assigned to opposite signs so that (*y*-*x*) has the same sign as *x* to indicate that *y* is encoded as the difference from *x*. In the second type ($x<0<y<z$ or $z<y<0<x$), including cases 7~12, two inputs (*x* and *y*) and one difference (*z*-*y*) are chosen. *x* and *y* have opposite signs and are indicated by the redundant bits. The difference z-y has the same sign as y to indicate that z is encoded as the difference from y. In this way, 12 cases are reduced to 6, and only 3 redundant bits are required.Table 3 shows the encoding and decoding functions and the assignment of the redundant bits. For the encoder, the inputs are *r*, *g*, and *b*, and the outputs are α, β, and γ. The redundant bits are z_0, z_1, and z_2. For example, in the first case $0<r<g<b$, *r* is the closest value to zero, so *r* is sent as is. The redundant bit z_0 is set to 1, indicating that no encoding/decoding is needed for the *r* channel. The *g* is encoded as *g*-*r*, which has the same sign as *g*, while *b* is encoded as *g*-*b*, which has the opposite sign. The z_1z_2 are set as zeros, meaning that decoding is needed, which in turn requires the comparison of their signs. Consider another example: In the 7th case $b<0<g<r$, the *b* and *g* are sent without encoding, while the *r* is encoded as *r*-*g*, which has the same sign as *g*. The $z_0z_1z_2$ are set as 011. The *r* can be recovered correctly based on the fact that *r*-*g* has the same sign as *g*.

Notice that the subtraction computation is performed in the extended range [-255,255]. After the spatial and chromatic encoding, if the resulted encoded data from any channel falls out of the range [-128,127], then all of the three channels can be encoded by the original TMDS algorithm (*TM*). This condition is indicated by the use of redundant bits equal to 111 and interpreted in the same way as the *E[8]* bit of the DVI. Optionally, the three original values can be sent without any encoding if it generates fewer transitions than the TM encoding.

4.3 Experimental results

Experiments are performed to evaluate the power savings of the chromatic encoding over the original DVI encoding. For fairness, some minor modifications are needed.

In the DVI implementation, since the *E[9]* bit does not affect the intra-word transition counts, it is set to zero, and therefore no inversion is applied. Because the decision between XOR and XNOR, which is indicated by *E[8]*, does not affect the transition counts, the *E[8]* bit is always set to zero. In the chromatic encoding implementation, the *E[9]* is set to zero as is the DVI. The three *E[8]* bits from the R, G, and B channels are used to carry the three redundant bits $z_0z_1z_2$.

Table 3: The chromatic encoding function.

#	Ranking	Encoding					Decoding		
		α	β	γ	$z_0z_1z_2$	if sign()	*r*	*g*	*b*
1	$0<r<g<b$; $b<g<r<0$	*r*	*g*-*r*	*g*-*b*	100	$\alpha=\beta$	α	α+β	α+β–γ
2	$0<r<b<g$; $g<b<r<0$	*r*	*b*-*g*	*b*-*r*	100	$\alpha=\gamma$	α	α+γ–β	α+γ
3	$0<g<b<r$; $r<b<g<0$	*b*-*r*	*g*	*b*-*g*	010	$\beta=\gamma$	β+γ–α	β	β+γ
4	$0<g<r<b$; $b<r<g<0$	*r*-*g*	*g*	*r*-*b*	010	$\beta=\alpha$	β+α	β	β+α–γ
5	$0<b<r<g$; $g<r<b<0$	*r*-*b*	*r*-*g*	*b*	001	$\gamma=\alpha$	γ+α	γ+α–β	γ
6	$0<b<g<r$; $r<g<b<0$	*g*-*r*	*g*-*b*	*b*	001	$\gamma=\beta$	γ+β–α	γ+β	γ
7	$b<0<g<r$; $r<g<0<b$	*r*-*g*	*g*	*b*	011	$\alpha=\beta$	β+α	β	γ
8	$g<0<b<r$; $r<b<0<g$	*r*-*b*	*g*	*b*	011	$\alpha=\gamma$	γ+α	β	γ
9	$r<0<b<g$; $g<b<0<r$	*r*	*g*-*b*	*b*	101	$\beta=\gamma$	α	β+γ	γ
10	$b<0<r<g$; $g<r<0<b$	*r*	*g*-*r*	*b*	101	$\beta=\alpha$	α	β+α	γ
11	$g<0<r<b$; $b<r<0<g$	*r*	*g*	*b*-*r*	110	$\gamma=\alpha$	α	β	γ+α
12	$r<0<g<b$; $b<g<0<r$	*r*	*g*	*b*-*g*	110	$\gamma=\beta$	α	β	γ+β
13	*overflow*	r_0	g_0	b_0	000	-	α	β	γ
14	*overflow*	*TM*(r_0)	*TM*(g_0)	*TM*(b_0)	111	-	*TM*(α)	*TM*(β)	*TM*(γ)

In the DVI implementation, since the *E[9]* bit does not affect the intra-word transition counts, it is set to zero, and therefore no inversion is applied. Because the decision between XOR and XNOR, which is indicated by *E[8]*, does not affect the transition counts, the *E[8]* bit is always set to zero. In the chromatic encoding implementation, the *E[9]* is set to zero as is the DVI. The three *E[8]* bits from the R, G, and B channels are used to carry the three redundant bits $z_0 z_1 z_2$.

Table 4: Transition savings.

	Spatial	Spatial+ Chromatic	Overflow
4.1.01	46.03%	54.36%	0
4.1.02	44.60%	54.70%	6
4.1.03	64.57%	72.83%	1
4.1.04	50.72%	60.83%	241
4.1.05	55.73%	62.96%	2
4.1.06	48.35%	57.61%	80
4.1.07	67.73%	73.20%	0
4.1.08	63.74%	69.98%	0

The tonal encoding reduces 54% to 73% of the transition counts. The overflow condition occurs rarely. Six of the eight images have none or very few pixels that overflow. It implies that the encoding algorithm reduces the magnitude of the pixel values effectively.

The eight benchmark images may not be representative enough to evaluate the tonal locality and chromatic encoding due to the limited sample size. Instead of still images, three video clips, which consist of hundreds of images, are used as a more general benchmark suite. The video clips were selected from different sources to present diversity. The *tiger* movie is a documentary on wild animals. The *wg* movie is a clay animation. The *final3* movie is a Japanese anime. The results are presented in Table 5. Up to 75% of the transition counts can be eliminated by using the spatial and chromatic encoding.

Table 5: Transition reduction by chromatic encoding.

Clip name	*wg*	*tiger*	*final3*
Source type	clay animation	camcorder	anime
Frame number	331	634	1018
Frame size	304*224	320*240	160*128
Spatial	57.17%	38.89%	49.56%
Spatial+Chromatic	75.55%	64.47%	73.63%

The proposed encoder has been implemented in VHDL and synthesized by the Synopsys Design Analyzer and TSMC 0.18 library. The power consumption from the simulation is 55.07mW at 1.8V 1.65GHz. The spatial encoder, chromatic encoder, and codebook consume 12.52mW, 38.97mW, and 1.04mW, respectively.

5 Conclusion

This paper has presented a low-power encoding technique, chromatic encoding, for the DVI, a digital serial video interface. We have proved that chromatic encoding reduces power consumption by minimizing the transition counts on the DVI. This technique relies on the notion of tonal locality, the observation that the signal differences between adjacent pixels in images follow a Gaussian distribution. Based on this observation, an optimal code assignment is performed to minimize the transition counts. Furthermore, it is found that the three color channels of the DVI may be reciprocally encoded to achieve even more power saving. The proposed technique requires only three redundant bits for each 24-bit pixel. Experimental results show up to a 75% transition reduction in the DVI.

Acknowledgements

We acknowledge Jianlin Liang, Changwoo Kang and Peng Rong for their help on profiling and implementation issues.

References

[1] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. VLSI Systems*, vol. 7, no. 2, pp. 212-221, 1999.

[2] L. Benini et al., "System-level power optimization of special purpose applications: The beach solution," *Proc. ACM/IEEE Int'l Symp. Low Power Electronics and Design (ISLPED-97)*, pp. 24-29, 1997.

[3] E. Musoll, T. Lang, and J. Cortadella. "Exploiting the locality of memory references to reduce the address bus energy," *Proc. ACM/IEEE Int'l Symp. Low Power Electronics and Design (ISLPED-97)*, pp. 202-207, 1997.

[4] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. VLSI Systems*, vol. 3, no. 1, pp. 49-58, 1995.

[5] Intel, *Intel Pentium 4 Processor with 512-KB L2 Cache on 0.13u Process at 2 GHz, 2.20 GHz, and 2.40 GHz Datasheet*, 2002.

[6] W. Kowalsky et al., "OLED matrix displays: technology and fundamentals," *Proc. First Int'l Conf. Polymers and Adhesives in Microelectronics and Photonics*, pp. 20-28, 2001.

[7] Video Electronics Standards Association. http://www.vesa.org.

[8] Digital Display Working Group, *Digital Visual Interface, V 1.0*, April 1999. http://www.ddwg.org.

[9] Extron Electronics, *Cable Products Guide*, http://www.extron.com.

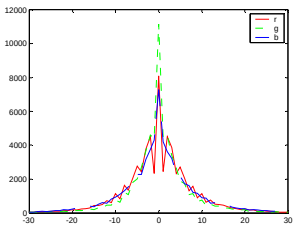
[10] Intel, *Intel PXA250 and PXA210 Application Processors Developer's Manual*, Feb. 2002.

[11] W. K. Pratt, *Digital Image Processing*, New York: John Wiley & Sons, Inc., 1991.

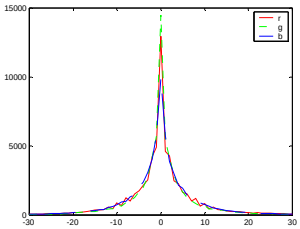
[12] A. G. Weber, "The USC-SIPI Image Database Version 5," *USC-SIPI Report #315*, Oct. 1997. http://sipi.usc.edu/services/database/Database.html.

[13] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Cambridge: The MIT Press, 1990.

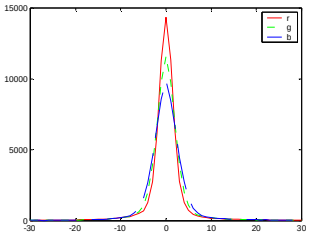
Appendix: The USID images and distributions of the pixel value differences



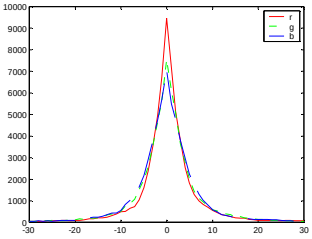
4.1.1



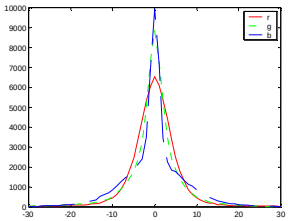
4.1.2



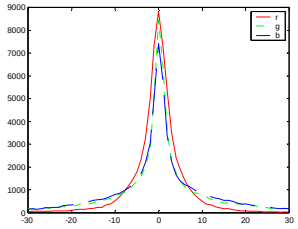
4.1.3



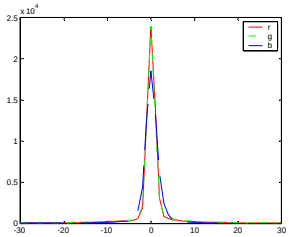
4.1.4



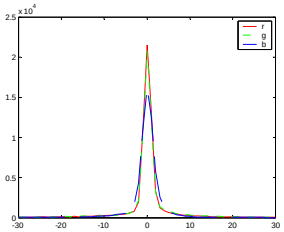
4.1.5



4.1.6



4.1.7



4.1.8