Single-Scale Retinex Using Digital Signal Processors

Glenn Hines
NASA Langley Research Center
MS 472,
Hampton, VA
23681 US
757 864-8139
glenn.d.hines@nasa.gov

Zia-ur Rahman
College of William & Mary
Dept. of Applied Science
Williamsburg, VA
23187 US
757 221-3479
zrahman@as.wm.edu

Daniel Jobson
NASA Langley Research Center
MS 473
Hampton, VA
23681 US
757 864-1521
daniel.j.jobson@nasa.gov

Glenn Woodell
NASA Langley Research Center
MS 473
Hampton, VA
23681 US
757 864-1510
glenn.a.woodell@nasa.gov

ABSTRACT

The Retinex is an image enhancement algorithm that improves the brightness, contrast and sharpness of an image. It performs a non-linear spatial/spectral transform that provides simultaneous dynamic range compression and color constancy. It has been used for a wide variety of applications ranging from aviation safety to general purpose photography. Many potential applications require the use of Retinex processing at video frame rates. This is difficult to achieve with general purpose processors because the algorithm contains a large number of complex computations and data transfers. In addition, many of these applications also constrain the potential architectures to embedded processors to save power, weight and cost. Thus we have focused on digital signal processors (DSPs) and field programmable gate arrays (FPGAs) as potential solutions for real-time Retinex processing. In previous efforts we attained a 21 (full) frame per second (fps) processing rate for the single-scale monochromatic Retinex with a TMS320C6711 DSP operating at 150 MHz. This was achieved after several significant code improvements and optimizations. Since then we have migrated our design to the slightly more powerful TMS320C6713 DSP and the fixed point TMS320DM642 DSP. In this paper we briefly discuss the Retinex algorithm, the performance of the algorithm executing on the TMS320C6713 and the TMS320DM642, and compare the results with the TMS320C6711.

Keywords: image enhancement, digital signal processing, retinex

1. INTRODUCTION

The Retinex is a general purpose image enhancement algorithm that is used to improve the contrast, brightness and perceived sharpness of images primarily through dynamic range compression. The algorithm also simultaneously provides color constant output thus it removes the effects caused by different illuminants on a scene. It synthesizes contrast enhancement and color constancy by performing a non-linear spatial/spectral transform that mimics traits seen in the human vision system. The original algorithm is based on a model of human vision's lightness and color constancy developed by Edward Land.[1] Jobson et al. extended the last version of Land's model[2,3] and have since added several improvements to the original version of the Retinex including the use of multiple scales,[3] color restoration,[3] and post-processing using white balance.[4]

The unique enhancement achieved by the Retinex lends the algorithm to numerous applications. The algorithm has been successfully applied to imagery from diverse fields such as medical radiography, underwater photography, and forensic investigations. It is being used to process visible and infrared imagery acquired to produce enhanced vision systems (EVS) for aviation safety,[5] and it is used to improve imagery obtained from unmanned aerial vehicles (UAV)s. Figure 1 is an example of a Retinex processed image for aviation safety. It is being studied for potential use in X-ray systems to improve homeland security. A large majority of consumer level photographs also benefit from

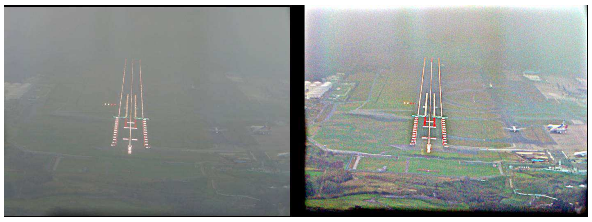

Figure 1. Original image of a runway on the left; Retinex enhanced image on the right. Note the improvement in visibility of the runway and surrounding areas.

Retinex processing and the algorithm is offered in the commercially available software package PhotoFlair by TruView Imaging and as an Adobe Photoshop plug-in.[6]

Computation of the Retinex algorithm involves performing a large number of complex operations and data transfers. For individual, smaller format images, standard general purpose computers provide sufficient processing power and reasonable performance. To apply the algorithm to images acquired at real-time video data rates of 15 to 30 frames per second (fps) requires the substantial increase in processing speed afforded by hardware performance. In addition, several potential applications limit the hardware solutions to low-power, low cost, embedded systems.

Several dedicated architectures and technologies exist that are potentially, a good fit for real-time Retinex processing. In our current implementation we have targeted digital signal processors (DSPs), in particular the Texas Instruments TMS320C6711, TMS320C6713 floating point processors and the DM642 fixed point processor. DSPs are inexpensive, relative easy to program and offer good performance for real-time applications. In addition they have specialized instructions, such as multiply-accumulate or bit reversal, and fast data transfer paths and mechanisms that facilitate high bandwidth image processing.

We recently developed the first near real-time digital implementation of the single-scale monochromatic Retinex.[7] A video frame rate of 21 fps was attained using a 150 MHz TI TMS320C6711 DSP evaluation module (DSK) with full-frame (256×256) processing. Standard NTSC video with a frame size of 640×480 was captured, scaled to 320×240, Retinex processed as a 256×256 image, and finally displayed on a standard VGA monitor. Video capture and display was performed on a daughter-card that interfaced to the C6711 evaluation DSK. Both the original and the Retinex processed images were displayed on the output for performance assessment and demonstration purposes.

The next stage of real-time Retinex processing is to increase the performance closer to 30 fps. The frame rate of 21 fps was only achieved after several significant code improvements and optimizations on the C6711 DSP.[7] This includes using 2-dimensional Direct Memory Access (DMA) transfers to improve column access of image data, merging algorithm components to maintain cache coherency, using cache-optimized FFT routines found in the DSP Library, and using double buffering schemes to overlap processing time with data transfer times.

It may be theoretically possible to slightly increase performance to greater than 21 fps on the C6711 by performing optimizations such as writing all of the code in hand-optimized assembly, using more in-line functions, or unrolling more loops by hand, but the benefit gained is minimal for the effort required. Thus to increase performance we have migrated our design to the slightly more powerful TMS320C6713 DSP and the fixed point TMS320DM642. We will now briefly describe the equations behind the Retinex algorithm, give an overview of the three processors, describe the testing environment, and discuss the results obtained for each processor.

2. RETINEX

The Retinex is a member of the class of center surround functions where each output value of the function is determined by the corresponding input value (center) and its neighborhood (surround). For the Retinex the center is defined as each pixel value and the surround is a Gaussian function. The mathematical form of the single-scale Retinex (SSR) is given by

$$R(x_1, x_2) = \alpha\big(\log(I(x_1, x_2)) - \log(I(x_1, x_2) * F(x_1, x_2))\big) - \beta$$

where I is the input image, R is the Retinex output image, log is the natural logarithm function, α is a scaling (gain) factor and β is an offset parameter. The "$*$" symbol represents convolution. F is a Gaussian filter (surround or kernel) defined by

$$F(x_1, x_2) = \kappa \exp[-(x_1^2 + x_2^2)/\sigma^2]$$

where σ is the standard deviation of the filter and controls the amount of spatial detail that is retained, and κ is a normalization factor that keeps the area under the Gaussian curve equal to 1. The α, β and σ parameters are determined empirically.

As is easily observed from the Retinex equation, the tallest processing pole is the convolution operation. Large Gaussian kernels, typically with σ ranging from 50 to 120, are normally used to produce good single-scale Retinex performance, thus spatial domain convolution would be extremely time consuming. We naturally turn to the well-known equivalence between convolution in the spatial domain and multiplication in the spatial-frequency domain[8, 9]

$$f(x, y) * g(x, y) \Leftrightarrow F(\mu, \nu)G(\mu, \nu)$$

where F and G are the spatial frequency domain representations of f and g respectively. We employ the 2-dimensional $M \times N$ forward and inverse Discrete Fourier Transforms (DFT),[9]

$$\mathcal{F}(\mu,\nu) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{x=0}^{N-1} f(x,y)\exp[-j2\pi(\mu x/M + \nu y/N)]$$

$$f(x,y) = \sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1} \mathcal{F}(\mu,\nu)\exp[j2\pi(\mu x/M + \nu y/N)],$$

to rewrite the Retinex equation as:

$$R(x_1,x_2) = \alpha(\log(I(x_1,x_2)) - \log[\mathcal{F}^{-1}(I'(\mu,\nu)F'(\mu,\nu))]) - \beta,$$

where $I'(\mu,\nu)$ and $F'(\mu,\nu)$ represent the DFTs of $I(x_1,x_2)$ and $F(x_1,x_2)$ respectively, and $\mathcal{F}^{-1}$ represents the inverse DFT. The DFTs are computed using the well known Fast Fourier Transform (FFTs)[8] and by processing the 2-dimensional image transform by applying 1-dimensional FFTs first to the rows, and then to the columns of the image. DSPs are known for their ability to rapidly compute FFTs and the associated bit reversal required so this efficiency maps well into the equation above.

3. PROCESSORS

We consider three TI DSPs for Retinex computation: the TMS320C6711, the TMS320C6713 and the TMS320DM642. The TMS320C6711 (C6711) DSP is a 32-bit floating point processor that performs up to 1200 million instructions per second (MIPS)/900 million floating point operations per second (MFLOPs) at a clock rate of 150 MHz (6.67 ns instruction cycle time). It is based on the advanced very-long-instruction-word (VLIW)[10] architecture developed by TI. A block diagram of the processor is shown in Figure 2.

The processor is divided into three main components: the CPU (or core), memory, and peripherals. Details on registers, functional units and peripherals have been discussed previously.[7, 10, 11] Note that the internal processor memory consists of a two-level cache[12] where the Level-1 program cache (L1P) is 4-KBytes and direct mapped and the Level-1 data cache (L1D) is 4-KBytes and 2-way set associative. The Level-2 (L2) memory is 64-KBytes and can be configured as local SRAM, cache or combinations of the two in 16-KByte increments.

The TMS320C6713 (C6713) is a 32-bit floating point processor that performs up to 1800 MIPS/1350 MFLOPS at a clock rate of 225 MHz (4.4 ns instruction cycle time). The processor architecture is very similar to the C6711 and code operating on one device should directly port over to the other. The primary differences in the two devices are outlined in TI reports.[13, 14] In particular, the C6713 has an extra 192-KBytes of SRAM in the L2 memory. This increases the total size of the L2 memory to 256-KBytes of which 64-KBytes can serve as cache or SRAM. The C6713 also has programmable event mapping for its 16 EDMA channels. Also, relevant to this discussion is the addition of a software-configurable Phase-Lock Loop (PLL) controller on the C6713 that provides different clock frequencies for the DSP core, peripherals and the external memory interface (EMIF).

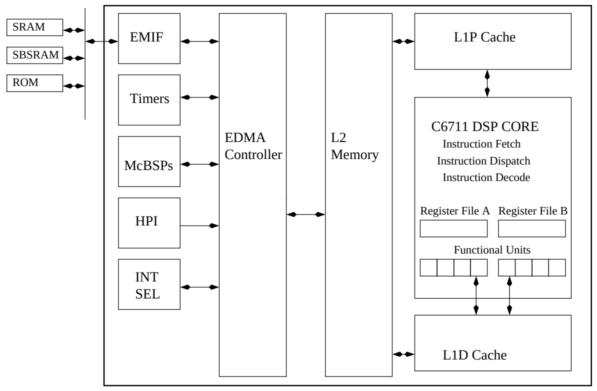

Figure 2. Basic 67x DSP Components: CPU, L1 Data and Program Caches, L2 memory (SRAM/Cache) and Peripherals

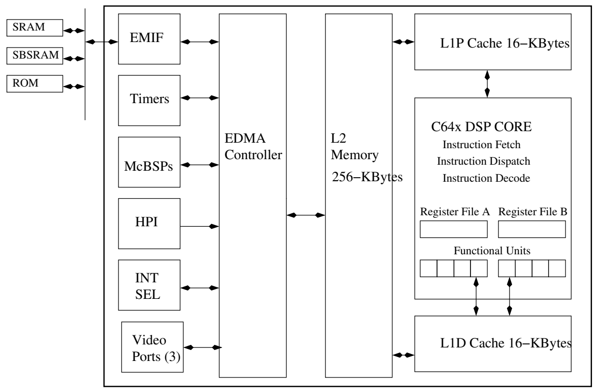

Figure 3. Basic DM642 Components: CPU, L1 Data and Program Caches, L2 memory and Peripherals. Note the additional Video Ports

The TMS320DM642 (DM642) is based on the same advanced VLIW architecture as the C6711 and C6713 but is structured differently than the other two processors. The DM642 is a 32-bit fixed-point processor that provides up to 4800 MIPS at a clock rate of 600 MHz (1.67 ns instruction cycle time). A block diagram of the processor is shown in Figure 3. Details on the processor can be found in TI reference papers.[15] We note in particular that the DM642 also has a two-level cache where the L1P is 16-KBytes and direct mapped, and the L1D is 16-KBytes and 2-way set associative. The L2 memory is 256-KBytes and can be configured as local SRAM, cache or combinations of the two in 16-KByte increments. The DM642 is in the family of TI's digital media processors where several media ports are built directly onto the chip. This includes three configurable video ports that can support either video capture and/or video display modes.

4. TEST ENVIRONMENT

Each of the DSPs is placed on an evaluation module called DSK for the C6711 and C6713, and EVM for the DM642. The C6711 DSK has 16-MBytes of 100 MHz SDRAM, peripheral connectors for daughterboard support and various ports and controllers.[16] The C6713 DSK is configured similarly except that it has 8-MBytes of SDRAM clocked at a default rate of 90 MHz. The DM642 EVM has 32-MBytes of SDRAM clocked at 133 MHz. There are also two video decoders and one video encoder to interface to the video ports of the chip and expansion connectors on the EVM board. The EVM communicates to a host computer through an external emulator via a JTAG connector. For the C6711 and C6713, video capture, display, and data formatting are performed by an imaging daughter-card (IDC).[17] The IDC contains an NTSC/PAL digital video decoder chip, an NTSC/PAL digital video encoder chip, a Xilinx FPGA and 16-Mbits of SDRAM for frame capture. The data formatting and buffering mechanisms performed by the IDC are described in TI reference documents.[18]

The test-bed for real-time Retinex video processing on the C6711 or C6713 consists of a camera, a monitor, the associated DSK, the IDC, a host PC and the associated software tools. The setup is the same for the DM642, except the IDC is not needed since the video ports are included on the chip and EVM. The camera generates NTSC/PAL composite video that is fed into the daughter-card/EVM. The RGB output of the daughter-card/EVM is fed into the CRT monitor for display. The host PC is a standard Pentium PC and is only used for code development and debugging. It is not part of the image processing chain. Figure 4 is an general outline of the system. The TI Code Composer Studio (CCS) tools are used for software development. This includes a C-compiler, assembler/optimizer, and a debugger for visibility into source code execution. A chip support library (CSL)[19] is used to configure and control on-chip peripherals and a DSP library (DSPLib)[20] is used to provide optimized FFTs.

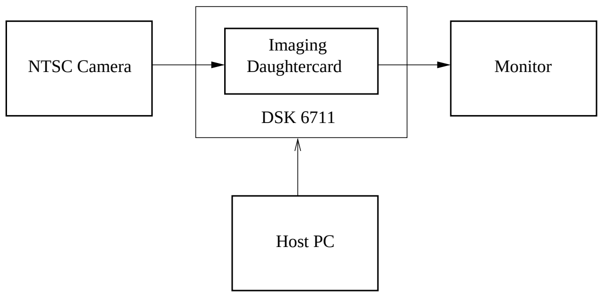

Figure 4. Test-bed — The PC only provides setup information to the DSK/DSP; after initiation, the DSP executes autonomously

General operation of the test-bed system is as follows. C code to perform the Retinex algorithm is written on the host PC. This code is compiled, assembled and linked into a common object file format (COFF) and downloaded from the host into the DSK/EVM. Execution of the algorithm is then initiated from the host. From this point on, the DSK/EVM operates independently of the host. The DSK, through the IDC, or the EVM captures video frames from the camera and resamples/averages the 640×480 pixel input image into a 320×240 sized image. This image is then cropped and padded to 256×256 pixels, Retinex processed and displayed adjacent to the unprocessed image for comparison. The final resizing to 256×256 is used to meet the power of two input size requirements of the FFT.

5. RESULTS

In previous efforts, we attained a 20.7 fps processing rate for the single-scale monochromatic Retinex executing on the C6711 operating at 150 MHz. To improve and compare performance we mapped the same code that executes on the C6711 onto the C6713. Considering the similarity in architectures this should provide a near linear increase in performance relative to the increase in clock speeds between the devices. Thus the performance should improve by 1.5 (225/150) so the expected performance should be close to 31 fps.

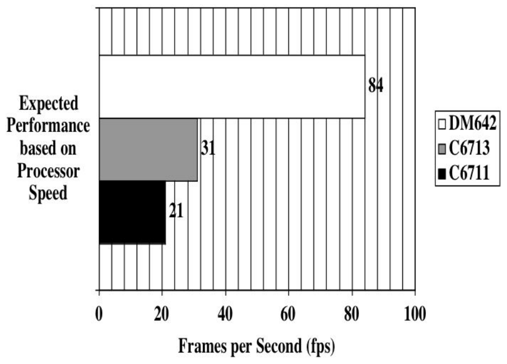

Figure 5. Expected DSP Performance based on MIPS for executing the Retinex Algorithm. Image size is 256 × 256

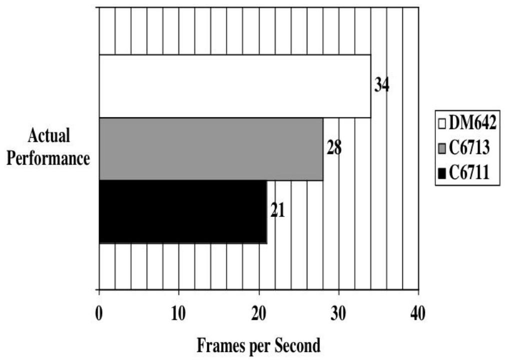

Figure 6. Actual Performance for the C6713 and DM642 was limited due to EMIF bandwidth.

Figure 5 shows the expected increase. Super-linear increases theoretically should be obtained because of the larger L2 memory, but all of the smaller buffers in the current implementation already reside in the 64-KByte L2 cache and the extra 192-KBytes of the C6713 are not enough to move any of the larger buffers into on-chip memory. After porting the code and moving the IDC from the C6711 to the C6713 the algorithm executed successfully. Figure 6 shows the actual increase in performance. The increase obtained is sub-linear due to the slower EMIF clock used for the SDRAMs of the C6713. The C6711 DSK uses a 100 MHz clock for the SDRAMS while the C6713 DSK uses a 90 MHz clock for the SDRAM thus limiting the data transfer rate between external memory and internal memory.

Next, we ported the Retinex algorithm to the DM642 platform. The DM642 uses different image capture and display drivers, DMA transfer buffers, and FFT algorithms than the C6711, but the core of the algorithm remains the same. Comparing DM642 MIPS with the C6711 shows a potential 4x increase in performance as seen in Figure 5. This does not take into account other factors that affect performance such as EMIF bus rates or the extra computations to handle fixed point arithmetic. Fixed-point arithmetic limits the dynamic range of the DM642 to $2^{31}-1$, while numbers are represented to an accuracy of 0.5×2^{-32} assuming Q0.31 format. For some portions of the algorithm the dynamic range is sufficient. For example, the input to a 256 point radix-4 FFT is processed in 4-stages where each stage gives 2 bits of growth. Our 8-bit input will then only grow to a maximum of 16 bits for one forward transform. Since we are generating a 2-dimensional Fourier Transform, a second 256-point FFT is also performed on the data. This increases the growth to 32 bits which still fits in a 32-bit integer data type. But this data is then further processed by the algorithm. The largest numbers from the FFT operation are on the order of 10^8. The smallest numbers from the normalized spatial frequency Gaussian function are truncated at 10^{-6}. Significant digits beyond this are truncated without affecting image quality. Thus we must process values on the order of 10^{14} well beyond the capability of 32-bit fixed point representation.

As a test case initially a floating point implementation of the Retinex algorithm was executed on the DM642 since the TI C compiler is able to automatically convert floating point operations, albeit very slowly. A data rate of 14.9 fps was achieved. After carefully balancing scaling and truncation tradeoffs a fixed point version of the algorithm was implemented. A processing rate of 30.31 fps was achieved. The limiting factor in performance is again the EMIF bus. With a faster processor, our algorithm performance is now driven by data transfer rates between external memory and internal memory exacerbated by the column access required for the 2-dimensional FFT. The default EMIF bus rate is set 133 MHz. We were able to update this bus rate to a maximum of 200 MHz by strapping the appropriate resistors onto the EVM module and changing memory access parameters. This increased performance to 34.1 fps as shown in Figure 6.

6. CONCLUSIONS

We have ported the Retinex algorithm that operates at 20.7 fps on a 150 MHz C6711 platform to a 225 MHz C6713 platform. Video images were captured using a NTSC camera and imaging daughter-card, single-scale, grayscale Retinex processed, and displayed using a standard VGA monitor. We obtained good (28 fps) but sub-linear performance from the C6713 platform due to the lower clock frequency used for external memory.

We also mapped the Retinex algorithm to the DM642 platform. The maximum performance obtained was 34.1 fps. This surpasses the real-time requirement of 30 fps. Again we found that the performance was primarily limited by data transfer bottleneck created by the EMIF bus. After using the DSP platforms, our next goal is to map the algorithm to a multi-FPGA system. The multiple processors in this system should provide the hardware performance required to achieve multiple scale, color Retinex processing and possibly multiple camera input processing. The reconfigurability of this system will give us a means to effectively tune the architecture for different application requirements. We also wish to consider platforms that will meet the needs for future space missions.

7. ACKNOWLEDGMENTS

The authors wish to thank the Systems Engineering Competency and the Aviation Safety and Security Program Office at NASA Langley for the funding which made this work possible. Dr. Rahman's work was supported under NASA cooperative agreements NCC-1-01030 and NNL04AA02A.

REFERENCES

1. E. Land, "An alternative technique for the computation of the designator in the retinex theory of color vision," in *Proceedings of the National Academy of Science*, **83**, pp. 3078–3080, 1986.
2. D. J. Jobson, Z. Rahman, and G. A. Woodell, "Properties and performance of a center/surround retinex," *IEEE Trans. on Image Processing* **6**, pp. 451–462, March 1997.
3. D. J. Jobson, Z. Rahman, and G. A. Woodell, "A multi-scale Retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing: Special Issue on Color Processing* **6**, pp. 965–976, July 1997.
4. Z. Rahman, D. Jobson, and G. Woodell, "Retinex processing for automatic image enhancement," in *Journal of Electronic Imaging*, **13, No. 1**, pp. 100–110, January 2004.
5. Rahman. see http://dragon.larc.nasa.gov/fog_haze for examples.
6. TruView. see http://www.truview.com.
7. G. Hines, Z. Rahman, D. Jobson, and G. Woodell, "Dsp implementation of the retinex image enhancement algorithm," in *Visual Information Processing XIII, Proceedings of SPIE 5438*, April 2004.
8. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, 1975.
9. R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, 1993.
10. Texas Instruments, "TMS320C6000 technical brief," Tech. Rep. SPRU197D, Texas Instruments, Dallas, Texas, February 1999.
11. Texas Instruments, "TMS320C6000 peripherals reference guide," Tech. Rep. SPRU190D, Texas Instruments, Dallas, Texas, February 2001.
12. Texas Instruments, "TMS320C621x/C671x dsp two-level internal memory reference guide," Tech. Rep. SPRU609A, Texas Instruments, Dallas, Texas, November 2003.
13. Texas Instruments, "Migrating from TMS320C6211B/C6711/C6711B and C6713 to TMS320C6713B," Tech. Rep. SPRA8561G, Texas Instruments, Dallas, Texas, March 2004.
14. Texas Instruments, "How to begin development today with the TMS320C6713 floating-point dsp," Tech. Rep. SPRA809A, Texas Instruments, Dallas, Texas, October 2002.
15. Texas Instruments, "TMS320C6000 technical brief," Tech. Rep. SPRS200E, Texas Instruments, Dallas, Texas, July 2002.
16. Texas Instruments, "TMS320C6000 imaging developer's kit (idk) user's guide," Tech. Rep. SPRU494a, Texas Instruments, Dallas, Texas, September 2001.
17. Texas Instruments, "TMS320C6000 imaging developer's kit (idk) programmer's guide," Tech. Rep. SPRU495A, Texas Instruments, Dallas, Texas, September 2001.
18. Texas Instruments, "TMS320C6000 imaging developer's kit (idk) video device driver user's guide," Tech. Rep. SPRU499, Texas Instruments, Dallas, Texas, December 2000.
19. Texas Instruments, "TMS320C6000 chip support library api user's guide," Tech. Rep. SPRU401E, Texas Instruments, Dallas, Texas, December 2002.
20. Texas Instruments, "TMS320C67x dsp library programmer's reference guide," Tech. Rep. SPRU657, Texas Instruments, Dallas, Texas, February 2003.