Real-Time Color Space Conversion for High Resolution Video

Klaus Gaedke, Jörn Jachalsky
Technicolor Research & Innovation, Germany
klaus.gaedke@technicolor.com

technicolor

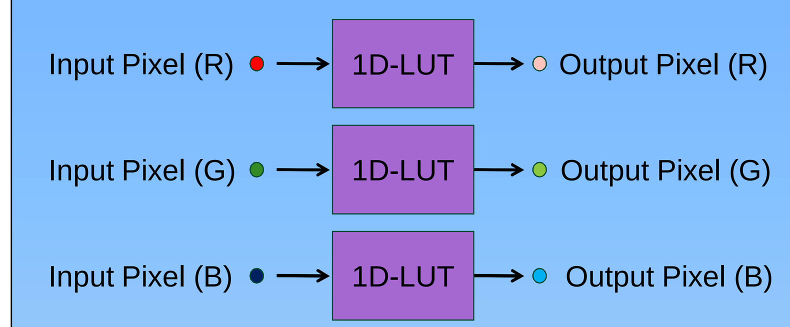
Color Space Conversion

Color space conversion or color correction is a widely used technique to adapt the color characteristics of video material to the display technology employed (e.g. CRT, LCD, projection) or to create a certain artistic look. As color correction often is an interactive task and colorists need a direct response, state-of-the-art real-time color correction systems for video are so far based on expensive dedicated hardware.

Look-Up-Tables for Color Space Conversion

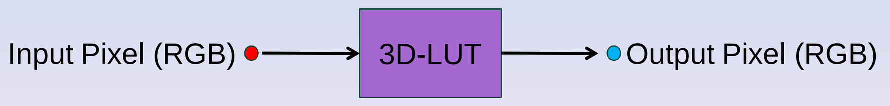
The use of Look-Up-Tables (LUTs) is a fairly simple, but very powerful approach for the implementation of color space conversion. Depending on the requirements for the conversion, either 1D- or 3D-LUTs have to be deployed.

Using 1D-LUTs, each component of an input pixel addresses one table. This results into a simple implementation with low memory requirements.



Color Resolution	Memory Space
8 bit	768 B
10 bit	3 kB
12 bit	12 kB

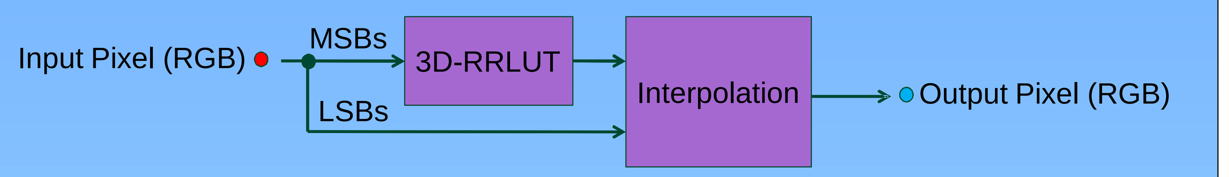
As color correction with 1D-LUTs can only be done for each color component separately, its usage is rather limited to very basic tasks like log-to-lin conversion or brightness control. For more sophisticated color correction tasks, 3D-LUTs have to be used instead. Here, all components of a pixel together build the address for the LUT. Thus, each color of the input color space can be mapped to any color of the output color space.



This approach can be applied to almost any color space conversion task, but the memory requirements are extreme.

Color Resolution	Memory Space
8 bit	50 MB
10 bit	4 GB
12 bit	309 GB

In order to reduce the memory requirements, a 3D-Reduced-Resolution-Look-Up-Table (3D-RRLUT) is combined with a unit interpolating the output values [1].



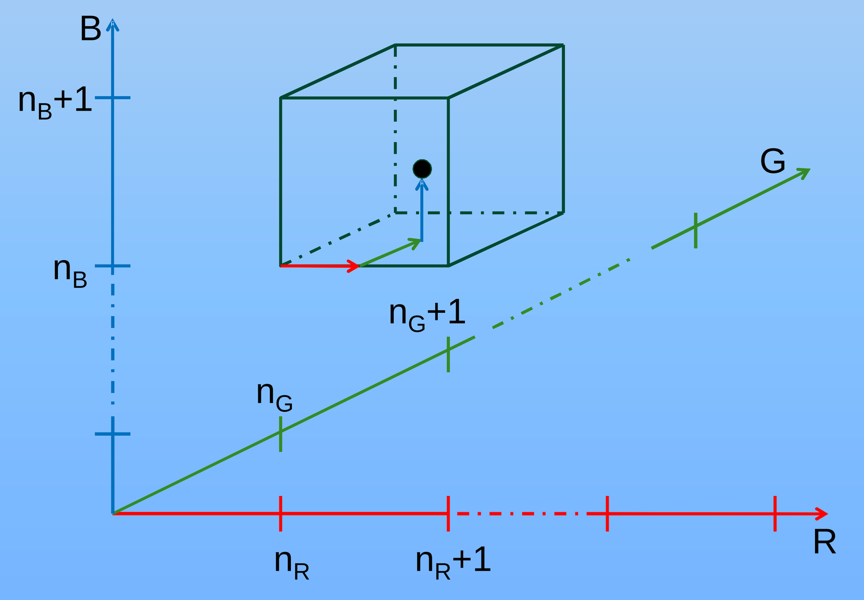
This approach reduces the memory requirements significantly. Even for 3D-RRLUTs with 12 bit color resolution and 129 entries for each color component, only 9.7 MB of memory are required.

Color Resolution	Number of Table Entries	Memory Space
8 bit	9x9x9	2.2 kB
	33x33x33	108 kB
	129x129x129	6.4 MB
10 bit	9x9x9	2.7 kB
	33x33x33	135 kB
	129x129x129	8 MB
12 bit	9x9x9	3.3 kB
	33x33x33	162 kB
	129x129x129	9.7 MB

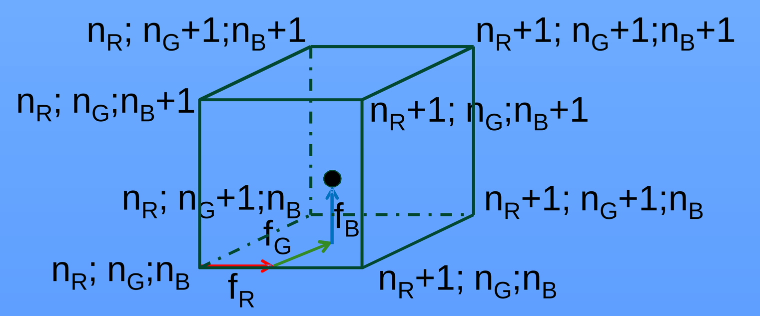
For the additional interpolation step often the efficient and precise Tetrahedral Interpolation is used.

Tetrahedral Interpolation

Tetrahedral Interpolation [1] is an efficient algorithm to interpolate the output values of a 3D-RRLUT.



Based on the MSBs $[n_R;n_G;n_B]$ of an input pixel, 8 table entries representing a cuboid in the output color space are addressed and read from the 3D-RRLUT.



This cuboid is decomposed into 6 tetrahedrons T1 to T6. Depending on the position $[f_R;f_G;f_B]$ of the input pixel within the cuboid, the corresponding tetrahedron is chosen and the related interpolation is calculated.

Tetrahedron	Condition	Interpolation Calculation
T1	$f_G \geq f_B \geq f_R$	$(1-f_G)\cdot[n_R;n_G;n_B] + (f_G-f_B)\cdot[n_R;n_G+1;n_B] + (f_B-f_R)\cdot[n_R;n_G+1;n_B+1] + (f_R)\cdot[n_R+1;n_G+1;n_B+1]$
T2	$f_B > f_R > f_G$	$(1-f_B)\cdot[n_R;n_G;n_B] + (f_B-f_R)\cdot[n_R;n_G;n_B+1] + (f_R-f_G)\cdot[n_R+1;n_G;n_B+1] + (f_G)\cdot[n_R+1;n_G+1;n_B+1]$
T3	$f_B > f_G \geq f_R$	$(1-f_B)\cdot[n_R;n_G;n_B] + (f_B-f_G)\cdot[n_R;n_G;n_B+1] + (f_G-f_R)\cdot[n_R;n_G+1;n_B+1] + (f_R)\cdot[n_R+1;n_G+1;n_B+1]$
T4	$f_R \geq f_G > f_B$	$(1-f_R)\cdot[n_R;n_G;n_B] + (f_R-f_G)\cdot[n_R+1;n_G;n_B] + (f_G-f_B)\cdot[n_R+1;n_G+1;n_B] + (f_B)\cdot[n_R+1;n_G+1;n_B+1]$
T5	$f_G > f_R \geq f_B$	$(1-f_G)\cdot[n_R;n_G;n_B] + (f_G-f_R)\cdot[n_R;n_G+1;n_B] + (f_R-f_B)\cdot[n_R+1;n_G+1;n_B] + (f_B)\cdot[n_R+1;n_G+1;n_B+1]$
T6	$f_R \geq f_B \geq f_G$	$(1-f_R)\cdot[n_R;n_G;n_B] + (f_R-f_B)\cdot[n_R+1;n_G;n_B] + (f_B-f_G)\cdot[n_R+1;n_G;n_B+1] + (f_G)\cdot[n_R+1;n_G+1;n_B+1]$

Implementation

The Tetrahedral Interpolation was implemented on 3 different parallel computing devices (IBM Cell B.E., NVIDIA Geforce GTS250, and NVIDIA Tesla C2050). Parallelization was done on pixel level. To avoid the significant performance loss caused by branches, always all 6 possible interpolations are calculated. The final result is computed with 3 fast select instructions.

OpenCL 1.0 was used as programming language for the GPUs. The Cell B.E. was programmed in C incl. Intrinsics.

Performance

The computation of the Tetrahedral Interpolation requires 250 arithmetic and logical operations for each pixel. The real-time processing requirements for common video formats range from 2.6 GOPS to 52.4 GOPS.

Image Size	Format	Frame or Field Rate	Operations
720x486	486i60	60 Hz interlaced	2.6 GOPS
1280x720	720p50	50 Hz progressive	11.5GOPS
1920x1080	1080p60	60 Hz progressive	31.1 GOPS
2048x1536	1536p25	25 Hz progressive	19.7GOPS
4096x2048	2048p25	25 Hz progressive	52.4 GOPS

The table below shows the performance of the Tetrahedral Interpolation on the 3 different parallel computing devices. Processing and transfer times are given per frame.

Computing Device	IBM CELL B.E.	Geforce GTS250	Tesla C2050
Host System	n.a.	Dual Socket XEON 5120	Dual Socket XEON W5590
Image Size	1920x1080	1920x1080	1920x1080
Processing Time on Device	16.6 ms	9 ms	1.5 ms
Transfer Time Host to Device	3.5 ms	4.2 ms	3 ms
Transfer Time Device to Host	3.5 ms	3.2 ms	3 ms
Bandwidth Host to Device	2.4 GB/s	2 GB/s	2.7 GB/s
Bandwidth Device to Host	2.4 GB/s	2.6 GB/s	2.7 GB/s
Processing Time +Transfer Time	16.6 ms[1]	16.4ms	7.5 ms
Optimization Level	Very high	Moderate	Moderate
Programming Language	C, many intrinsics	OpenCL 1.0	OpenCL 1.0
Avg. Performance	31.2 GOPS	31.6 GOPS	69.1 GOPS
Peak Performance	31.2 GOPS	57.6GOPS	345.6 GOPS
Processing Time on Host CPU only	n.a.	820 ms	767 ms
Speed-Up Device vs. CPU only	n.a.	50	102

[1]The Cell B.E. System supports parallel data transfer and computation

Conclusion

- Tetrahedral Interpolation performs very well on GPUs
  - High performance gain compared to CPUs
  - High utilization of computational resources on GPU
- Single low-cost GTS250 keeps pace with Cell B.E. and is sufficient for real-time color correction of 1080p60
- Single high-end Tesla C2050 supports real-time color correction up to the 2048p25 format

GPUs are ready to replace expensive dedicated hardware for real-time color space conversion

References: [1] H.-S. Lee, K.-S. Kim, D. Han, „A Real Time Color Gamut Mapping Using Tetrahedral Interpolation for Digital TV Color Reproduction Enhancement", IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, May 2009