3D LUTの基礎と HDRでの活用

2017年7月13日、14日
@アストロデザイン株式会社PRIVATE SHOW

WOWOWエンタテインメント株式会社　内田充洋

本セミナーの内容と目的

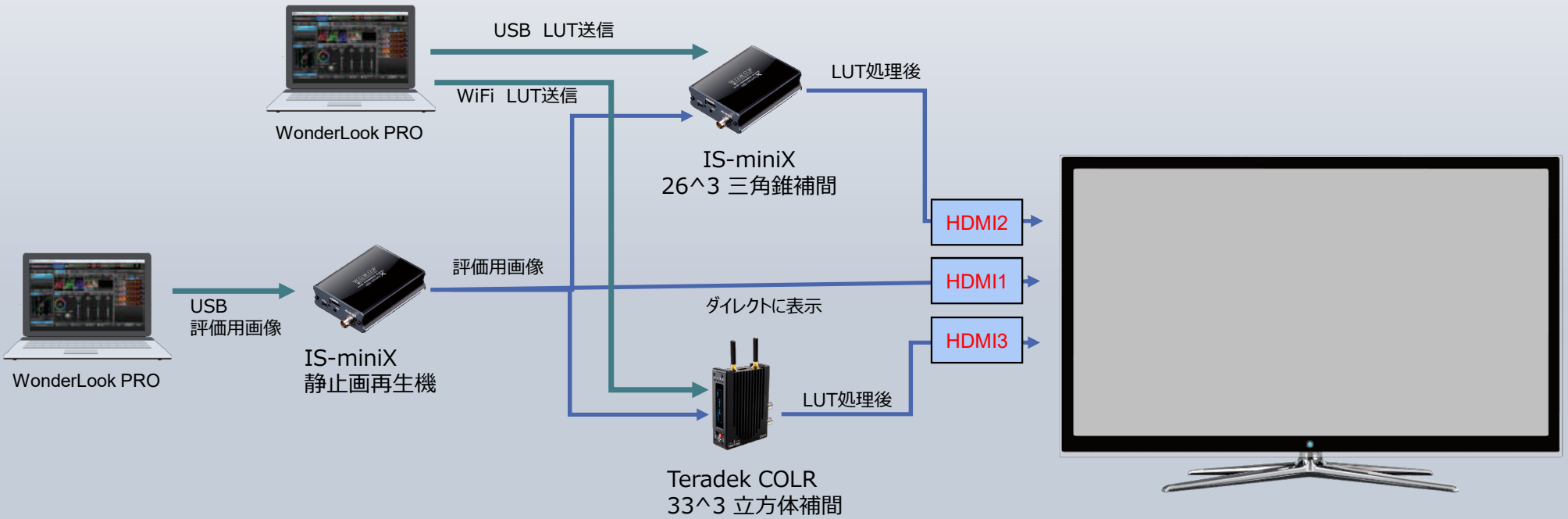
- 「色を変換する」とは？
- 色を変換する手段：計算式とLUTの２種の方法
- LUTとは何か：1DLUT / 3DLUT
- LUTの精度：グリッド数、補間方法、ShaperLUT
- LUTのハンドリング
- HDRでの活用

下記の疑問に答えられるようになることを目標とします。

・LUTとはどんなものか。精度に与える要因は何か。使用にあたってどのような注意が必要か。
・HDRで必要になる色変換はどのようなものがあるか。LUTはどのように活用すべきか。

デモ用システム構成図

戻る



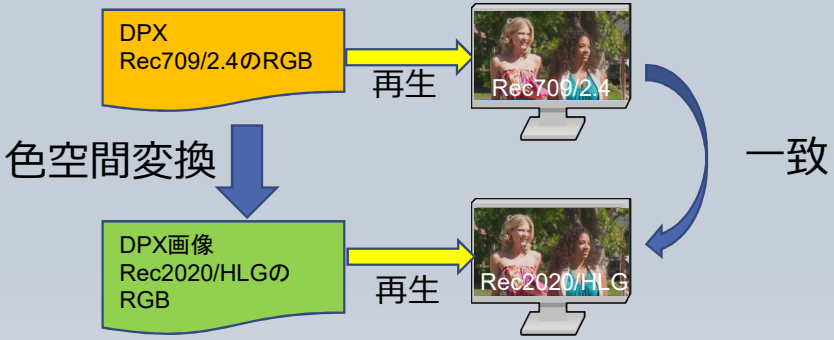
色を変換する

- 色を変換する目的は下記２種に分類できる。

異なる色空間に変換する(色空間変換)

- Rec709/2.4 → Rec2020/HLG
- LogC → SLOG3/SGAMUT3

色自体に変更を加えるわけではない。
RGBの表現方法を変えているだけ。
変換後も、映像としては同一。
比較的単純なものが多い。



色そのものを変換する（レンダリング）

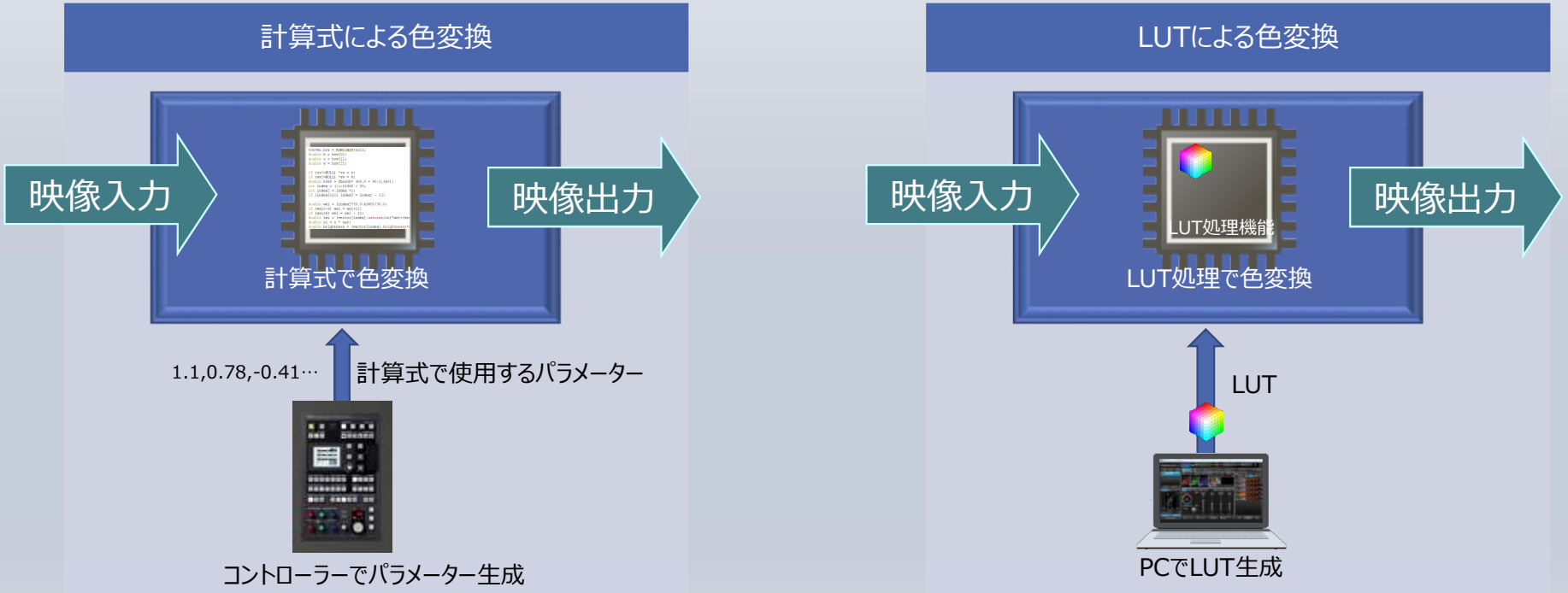
- カラコレ
- ホワイトバランス調整
- 本格的グレーディング

変換後は異なる映像となる。
かなり複雑な変換が含まれることもある。

色そのものを変換
（レンダリング）

ベースバンド映像の色をリアルタイムに変換する

- リアルタイムに色を変換するためには十分な高速性を確保する必要がある。
- 下記、２種の方法に分類できる。



計算式による色変換

メリット

- 誤差が極めて少ない変換
- 複数の変換をカスケード可能
- 逆変換も高精度で可能

デメリット

- 必要なすべての計算式をハードウエアに実装しておく必要がある
- 機能追加は容易ではない
- 複雑な計算や条件分岐が多い色変換など、リアルタイムに処理できないものもある

カメラ内の色変換
(LUTを使用できる機種有り)

FUJIFILM IS-100
(一部LUT使用)

SONY HDRC-4000

朋栄 FA-9500

AJA FS-HDR

LUTによる色変換

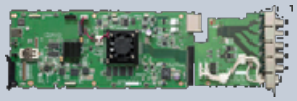
メリット

- **ハードウエアの変更なし**に、どんな色変換でも実現可能（色変換の自由度高い）
- どんな複雑な色変換も**処理時間一定**
- システム間**互換性が高い**(LUT交換による)

デメリット

- (補間する場合)**誤差が必ず発生**する
- LUT作成（更新）には必ず**PC/ソフトウエアが必要**
- 複数の変換のカスケードは誤差を蓄積
- 逆変換は不可、または大幅に精度劣化


IS-miniX


ASTRO SB-4024


FSI BoxIO


Teradek COLR

LUTを活用するには、デメリットに留意し、**誤差が**許容範囲以下になるようなワークフローを設計する必要がある。

計算式をハードウエアに実装可能な色変換

色変換 カテゴリ	変換の 複雑さ	例	コメント
色空間変換	シンプルな計算式	・Rec709→Rec2020 ・SDR→HDR	誤差も少なく最適
色空間変換 +シンプルなマッピング*	やや複雑な計算式	・HDR→SDR(mapping) ・Rec2020→Rec709(mapping)	採用可能なアルゴリズムは限られる。 いったん実装するとさらなる改良は困難。
レンダリング （色変換）	シンプルな計算式	・CCUによるカラコレ	入出力の色空間が限定され、クローズドなシステムとなる。

*マッピングについては後述

・特定用途で機能拡張の必要性がなく、計算式で実装可能な色変換は、計算式によるハードウエア化が適している。
・外部からパラメーターを供給するシステムも可能だが、クローズドなシステムとなってしまう。（汎用性なし）

計算式では実装が難しい色変換:3DLUTでのみ実現可能

色変換カテゴリ	内容	コメント
カメラのプロファイル測定に基づく変換	カメラの実測値を使ってSLOG3/SGAMUT3に変換する	1DLUT+3x3MTXでのモデル化は可能だが、誤差を低減するには実測値を使った非線形な変換が必要。
フィルムエミュレーション	プリントフィルムやネガフィルムの特性をエミュレートする色変換	フィルムの実測値を使って変換を定義しなければならない。
ACES RRT	デジタルシネマをターゲットとした標準の絵作りを定義	CTLというスクリプトで表現されているがかなり複雑であり、ハードウエアへの実装は困難
モニタキャリブレーション	モニタの誤差を補正し、目標となる色空間の定義通りの色再現を実現する	モニタの色を実測したデータに基づき、補正すべき変換を定義する。

・上記の変換は事実上3DLUTでしか実現できない。
・LUTをパラメーター化することで完全な汎用システムとなる。
・LUT処理は欠点があり注意が必要な部分もあるが、うまく活用することで柔軟なワークフローを構築できる。

LUTとは何か

LUTの基礎について考える

LUTの定義

・LUT(Look up Table) そのものは、入力の数値と出力の数値の対応関係を一覧表にしたもの
・色変換で使用されるLUTには、いくつかのルールがある

項目	説明
入力色空間	・ファイルには記載されないが、必ず**対象としている色空間が存在**する **・入力の数値範囲は有限であることに注意。（最小値、最大値が必ず存在）** ・テーブルでは入力側の数値が省略される場合が多い ・その場合は、0-1の範囲、刻みは等間隔（有限である） ・Shaper LUTという前処理が定義されている場合もある。
格子点数	テーブルの要素数。1DLUTの場合は、格子点数=テーブルの行数
出力色空間	ファイルには記載されないが、必ず**対象としている色空間が存在**する

LUTの例

入力値	出力値
0	5
1	6
2	8
3	10
79	49
80	50
81	51
119	118
120	120
121	122
139	239
140	240
141	241
252	253
253	254
254	255
255	255

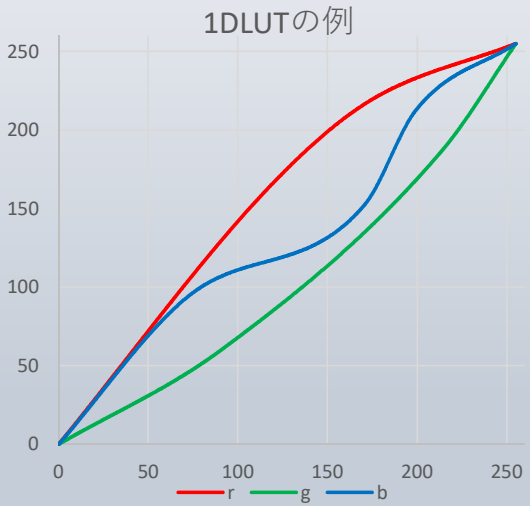
1DLUT(1次元LUT)で表現できること

【入出力色空間】
8bitのsRGB(0-255の整数)：格子点数256

$$R' = f_R(R)$$
$$G' = f_G(G)$$
$$B' = f_B(B)$$

1DLUTで表現可能

R用 LUT		G用 LUT		B用 LUT		
入力値	出力値	入力値	出力値	入力値	出力値	
0	5	0	5	0	5	
1	6	1	6	1	6	
2	8	2	8	2	8	
3	10	3	10	3	10	
79	49	79	49	79	49	
80	50	80	50	80	50	B
81	51	81	51	81	51	
119	118	119	118	119	118	
120	120	120	120	120	120	C
121	122	121	122	121	122	
139	239	139	239	139	239	
140	240	140	240	140	240	A
141	241	141	241	141	241	
252	253	252	253	252	253	
253	254	253	254	253	254	
254	255	254	255	254	255	
255	255	255	255	255	255	



RGBの計算式を独立に表現可能なもの

- ASC-CDL(Saturation以外)
- RGBトーンカーブ調整
- ホワイトバランス

- 出現可能性のある入力範囲をすべてカバーしておく必要がある（上記例だと0～255)
- RGBそれぞれ異なるテーブル(1DLUT)を用いることができる

3DLUTでしか表現できないこと

$$(R', G', B') = f\,(R, G, B)$$

RGBの計算式が分離できない場合

$$R' = \alpha(0.2627R + 0.6780G + 0.0593B)^{\gamma-1}R + \beta$$
$$G' = \alpha(0.2627R + 0.6780G + 0.0593B)^{\gamma-1}G + \beta$$
$$B' = \alpha(0.2627R + 0.6780G + 0.0593B)^{\gamma-1}B + \beta$$

上記はHLGのOOTFの一部の計算式
R,G,Bが分離できず、1DLUTでは表現できない
R’を求めるためには、R,G,B全てが必要

【入出力色空間】
8bitのsRGB(0-255の整数)

Rin	Gin	Bin	Rout	Gout	Bout
0	0	0	0	0	0
0	0	1	1	2	3
0	0	2	2	3	4
0	0	254	1	2	253
0	0	255	1	2	254
0	1	0	2	4	1
0	1	1	2	4	5
120	140	120	120	140	120
140	140	140	240	240	240
254	255	255	250	253	253
255	0	0	253	5	6
255	0	1	254	5	7
255	255	254	255	255	254
255	255	255	255	255	255

格子点数は256
テーブルの行数は256x256x256 = 16,777,216行！

・RGB3つの値をセットで入力して、初めてRGB３つの出力が得られるテーブル
・R,GまたはBの単独の入力はできない。

2DLUTで確認する　(RとGのみで色が表せる世界)

計算式

$$R' = 6R + 4G$$

$$G' = 4R + 6G$$

R'を求めるためにRとGが必要

2DLUT

入力：0～3の整数　出力：0～30の整数
格子点数：4

R'	R=0	R=1	R=2	R=3
G=0	0	6	12	18
G=1	4	10	16	22
G=2	8	14	20	26
G=3	12	18	24	30

G'	R=0	R=1	R=2	R=3
G=0	0	4	8	12
G=1	6	10	14	18
G=2	12	16	20	24
G=3	18	22	26	30

R→R'の入出力関係は、R'=f(R)で表現できず、Gにも依存する。

・RとGの1DLUT　2本では表現できないことを確認しよう
・この例を3DLUTに拡張するとどうなるか考えてみよう

3DLUTは巨大

入出力色空間が10bitのRec709：10bit(0-1023)すべての組み合わせのテーブルを用意する(格子点数1024)

1DLUT	2DLUT	3DLUT
• 1024x3 • 12KB • 1Dのサイズを1とすると	• 1024 x 1024 • 4.2MB • 1Dの350倍	• 1024 x 1024 x 1024 • 4.1TB • 1Dの35万倍

・3DLUTですべての組み合わせのテーブルを準備することは事実上不可能
・10bit整数でなく浮動小数点入力の場合、そもそもすべての組み合わせ自体存在しない

3DLUTは間引いたテーブルを補間して使う

3DLUT 間引き無し	3DLUT 1/8に間引き	3DLUT 1/16に間引き	3DLUT 1/32に間引き
• 格子点 1024 • 10億行 • 4.1TB • 1Dの35万倍	• 格子点 128 • 210万行 • 8.2MB • 1Dの683倍	• 格子点 64 • 262,144行 • 1.0MB • 1Dの85倍	• 格子点 32 • 32,768行 • 131KB • 1Dの13倍

・間引きによるサイズ削減効果は絶大。
・格子点数64ぐらいであれば、ハードウエアやソフトウエアでも取り扱うことが現実的なサイズとなる

間引きの例を見てみよう

R’	R=0	R=1	R=2	R=3
G=0	0	6	12	18
G=1	4	10	16	22
G=2	8	14	20	26
G=3	12	18	24	30

間引き無しの2Dテーブル(格子点数4)

R’	R=0	R=1	R=2	R=3
G=0	0			18
G=1				
G=2				
G=3	12			30

¼に間引いた2Dテーブル(格子点数2)

・データ量は4/16=1/4に削減できた。
・実際に使用するときに、空欄の値を推定しなければならない。→「補間」により推定を行う。

補間アルゴリズムによる違い

R’	R=0	R=1	R=2
G=0	0	18	30
G=1	4	10	15
G=2	8	14	20

間引き無しの2Dテーブル

R’	R=0	R=1	R=2
G=0	0		30
G=1		補間で推定	
G=2	8		20

2/3間引き

□ のデータを推定してみる

立方体補間

$$(0+30+8+20)/4=14.5$$

三角錐補間

$$(0+20)/2+8*0=10$$

$$(0+20)/2+30*0=10$$

・補間アルゴリズムにより結果が異なる。
・三角錐補間の方が軸外の極端な画素の影響を受けにくい
・R=1,G=0の画素はどうやっても推定不可能。入力側の端の色は補間誤差が出やすい。

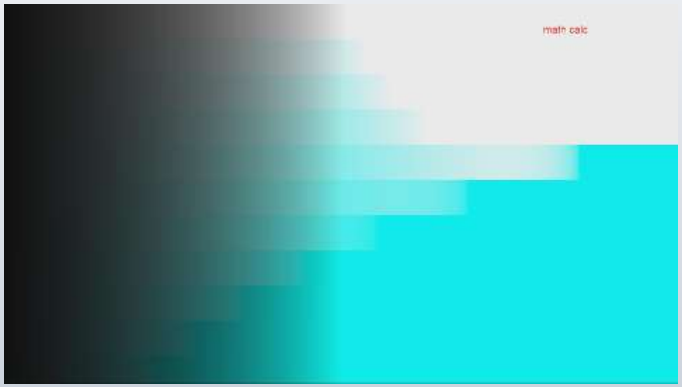
格子点の数と補間アルゴリズムによる結果の比較（デモ）

デモシステム図へ

- 使用した変換
  - Rec2020/ST-2084からSDRのRec709/2.4
- 格子点数の比較
  - 計算（正解値）、立方体補間で17^3、33^3、65^3、129^3
- 補間アルゴリズムの比較
  - IS-miniの三角錐補間(26^3)と立方体補間の各格子点数の比較

格子点の数によるバンディングと誤差の変化(立方体補間)

入力画像



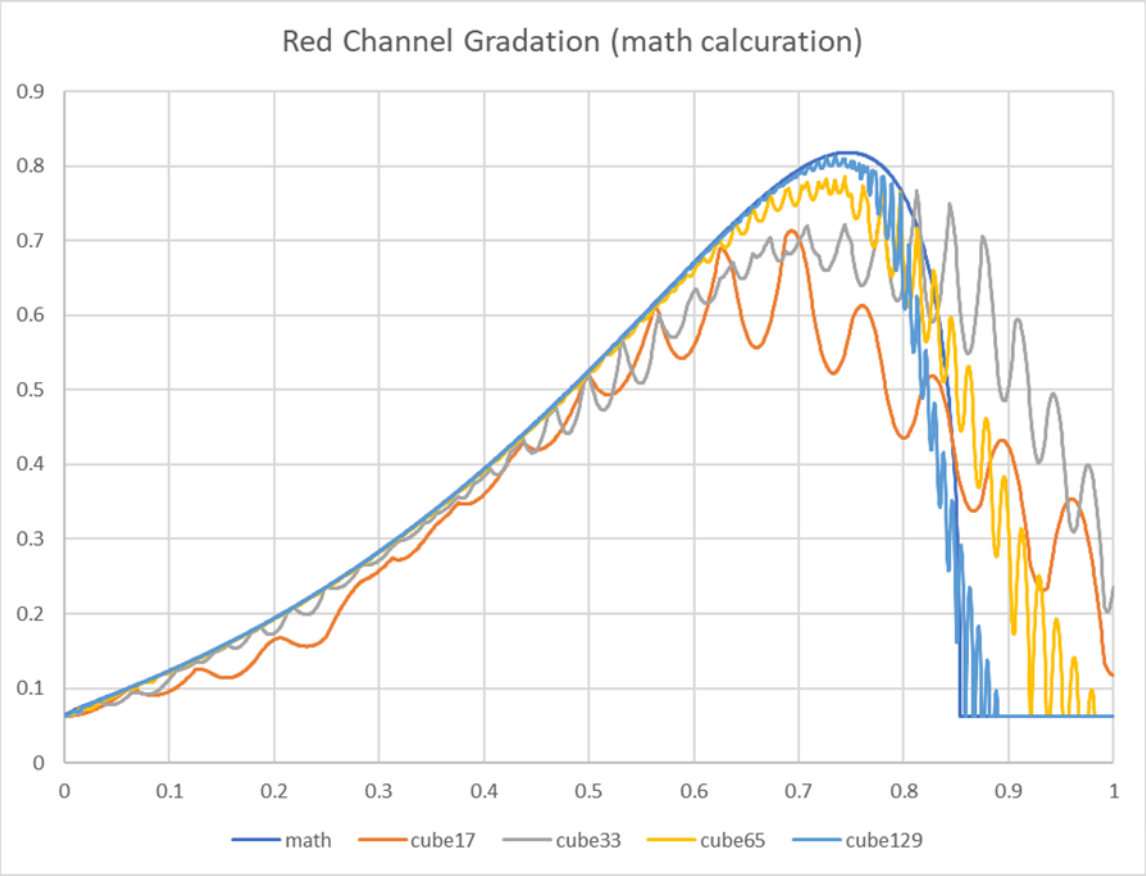
計算式による処理結果（正解画像）

格子点数１７

格子点数３３

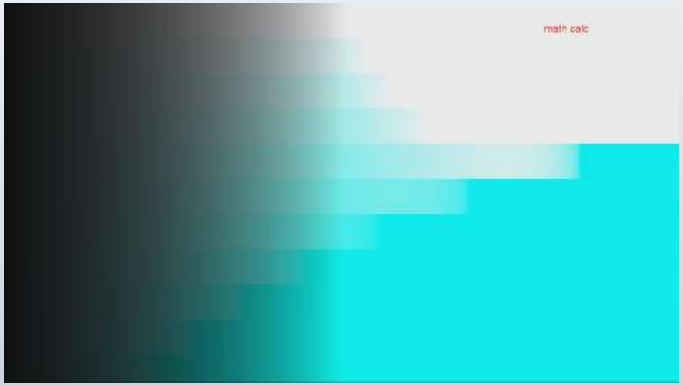
格子点数６５

格子点の数によるバンディングと誤差の変化(立方体補間)

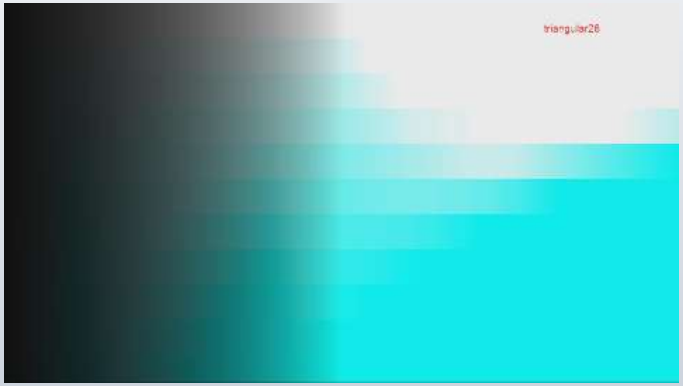


- 格子点の数を増やすとバンディングの振幅が小さく、周波数が高くなり、映像としては目立ちにくくなる。最大の129でもゼロにはならない。
- 計算との誤差も格子点の数で大きく変化。129でも若干の誤差は残る。

IS-miniの三角錐補間の実力



計算式による処理結果（正解画像）



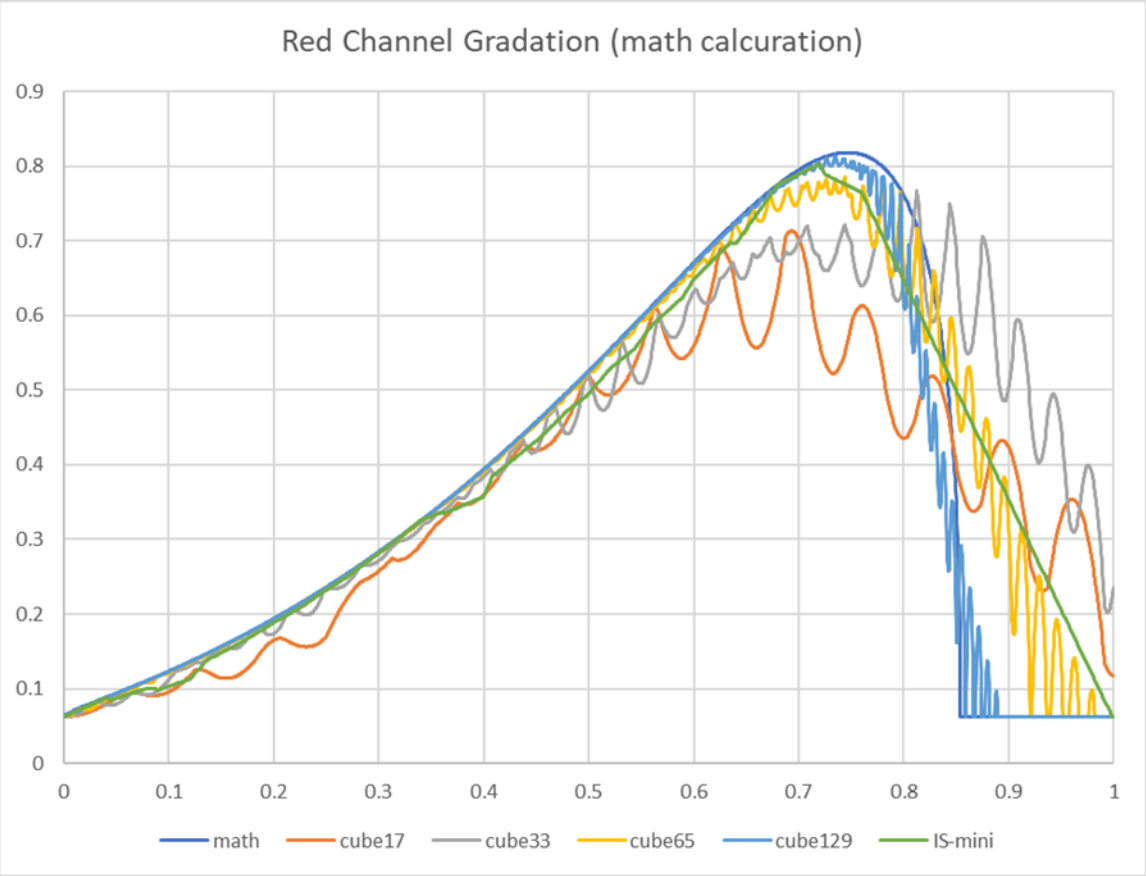
IS-mini 格子点数26(三角錐補間)

格子点数１７(立方体補間)

格子点数３３(立方体補間)

格子点数６５(立方体補間)

IS-miniの三角錐補間の実力



- IS-miniは格子点26で三角錐補間。
- バンディングはほとんど発生しない。
- 誤差は立方体補間の格子点65と同等レベル。
- 三角錐補間は、格子点の数で倍以上の立方体補間と同等の実力を持つことがわかる。

「LUTとは何か」のまとめ

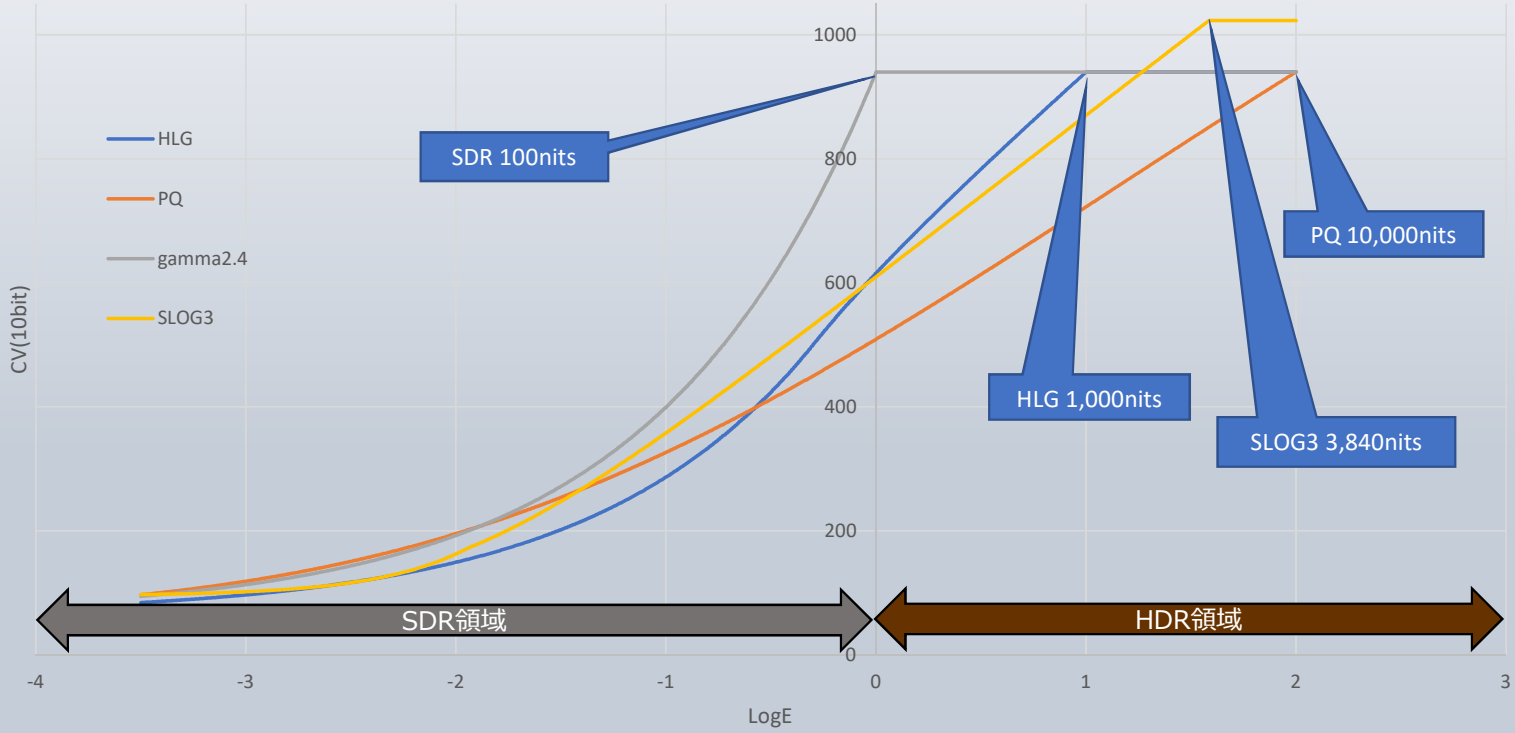
- 色変換の方法の一つ。「計算式」に比べ、「誤差」が発生する、というデメリットがあるが、自由度の高さ、拡張性の高さ、のメリットを生かした活用が進んでいる。
- LUTの「誤差」に関しては、「格子点の数」と「補間アルゴリズム」が二大要因。
- 「格子点」の数はなるべく多い方が良い。最低でも30前後、可能であれば65を選択すること。
- 「補間アルゴリズム」は可能であれば三角錐補間を選択すること。LUTBOXでは現状IS-miniだけだが、グレーディングソフトでは対応しているものもある。

※ASTRODESIGN社SB-4024は7月末のファームウエアで三角錐補間に対応予定

HDRでの活用

HDR制作の現場でLUTを活用してみる

各種HDRのダイナミックレンジ

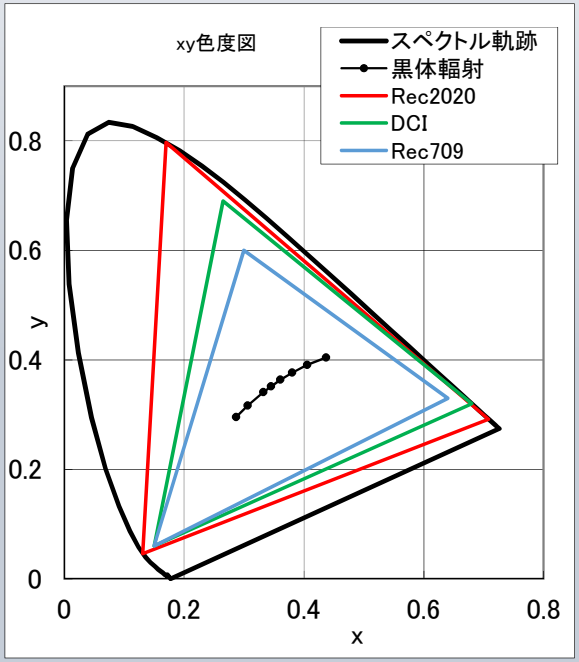


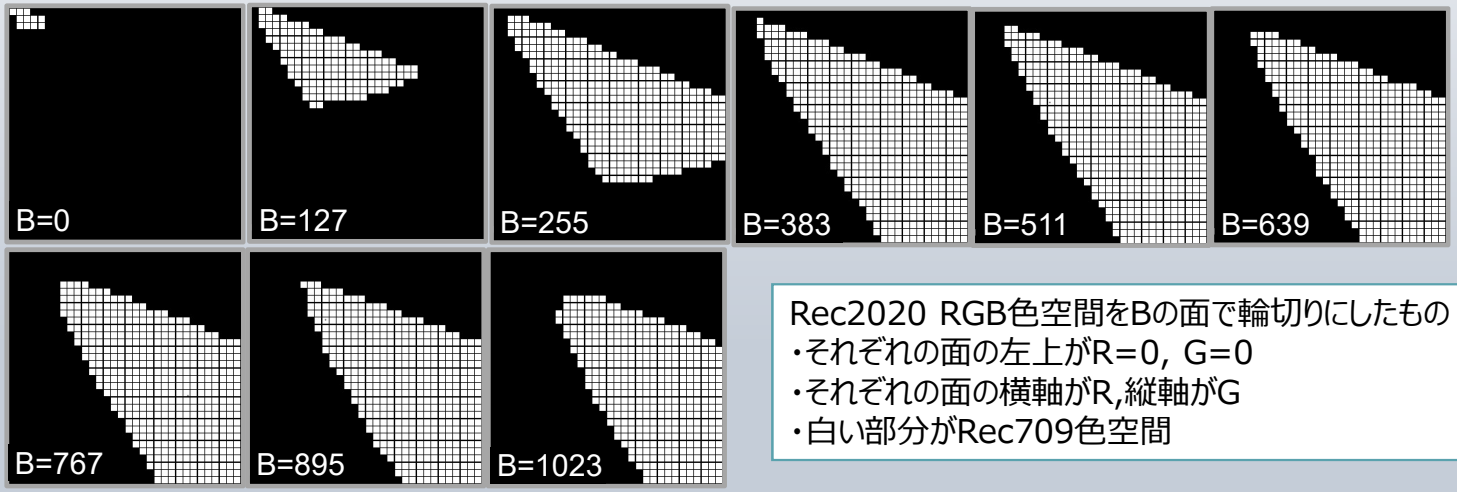
・HDRでも規格によりダイナミックレンジに違いがある
・ワークフロー構築時に、リアルタイムの相互変換が必要になる。

HDRの色空間：Rec2020

Rec2020 10bit RGB 100nits：Rec709 10bit RGB 100nits=　100：36.8

(Rec2020RGB 100nits 空間上での体積比)





Rec2020 RGB色空間をBの面で輪切りにしたもの
・それぞれの面の左上がR=0, G=0
・それぞれの面の横軸がR,縦軸がG
・白い部分がRec709色空間

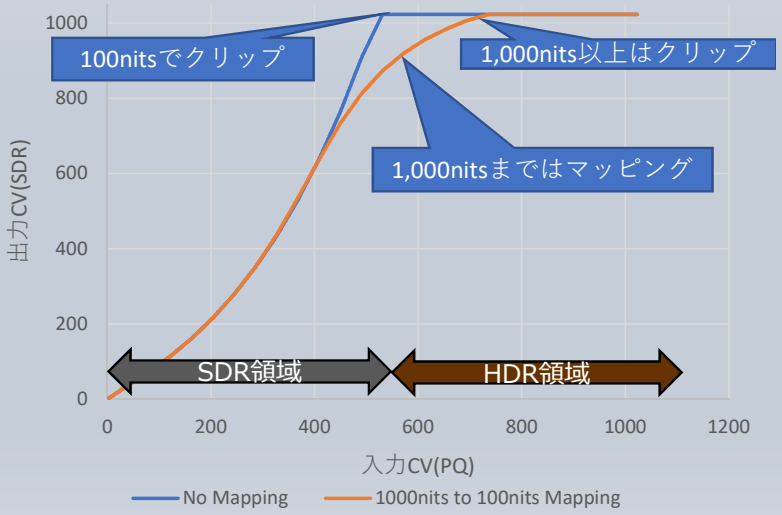
・Rec2020はRec709に対して3倍弱の色数を表現可能

広い色空間から狭い色空間への変換：マッピングとは

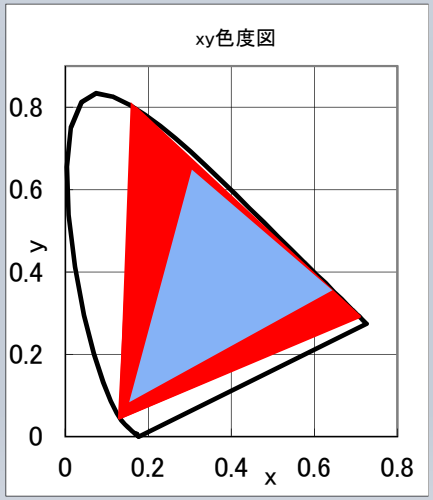
- 外側の色の処理方法で大きく２種類（クリッピングとマッピング）の方法に分類できる。どちらもメリット・デメリットがある。

名称	変換内容	メリット	デメリット
クリッピング	出力側の外側の色は、ディテールを切り捨て、出力側最大値を割り当てる。	内側の色が影響を受けずに保持される。	外側の色のディテールが失われる。
マッピング	出力側の外側の色のディテールも出力側で表現できるように、内側に丸め込む	外側の色のディテールがある程度表現できる。	内側の色も変化してしまう。

トーンカーブのマッピング
HDR(PQ) to SDR



色域のマッピング

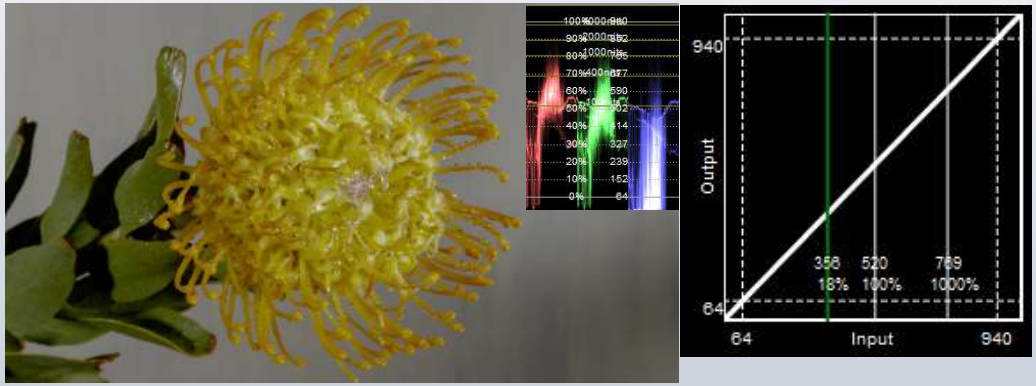


赤がRec2020,水色がRec709の色域

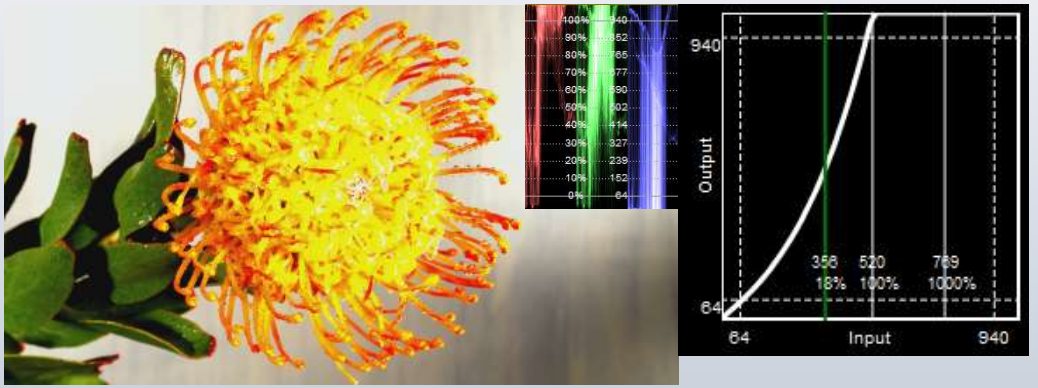
Rec2020からRec709へのマッピングは赤の領域を水色の領域内に収めること。種々の方法があるが、色相保持で彩度を落としながら内側に収める方法が一般的。

マッピングの具体例：トーンマッピング

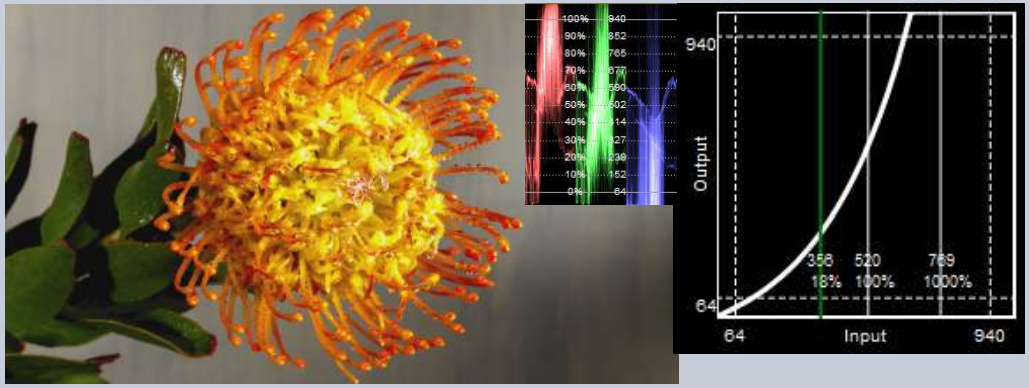
デモシステム図へ



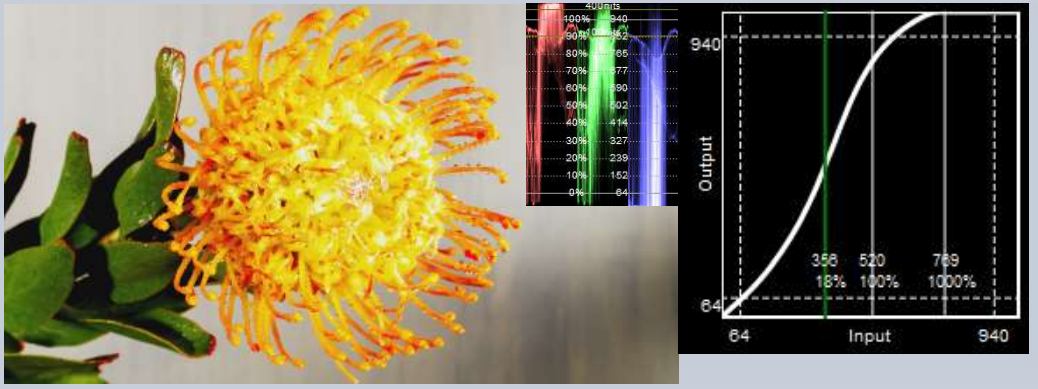
Rec2020/PQ



Rec2020/PQ→Rec709/2.4 Clipping



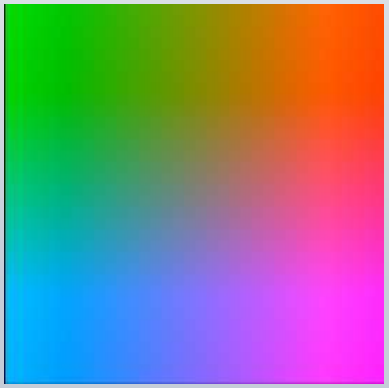
Rec2020/PQ→Rec709/2.4 -2.0EV



Rec2020/PQ→Rec709/2.4 Tone Mapping

画像はASTRODESIGN社8KHDR画像集“VT-7009”より

マッピングの具体例：ガマットマッピング

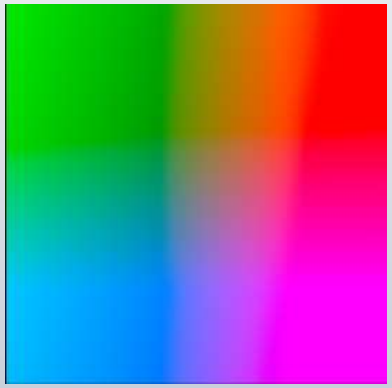


Rec2020 CrCb at Y=512

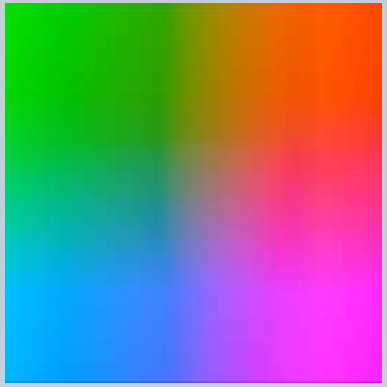
Clipping to Rec709

Mapping to Rec709

Rec709 no mapping

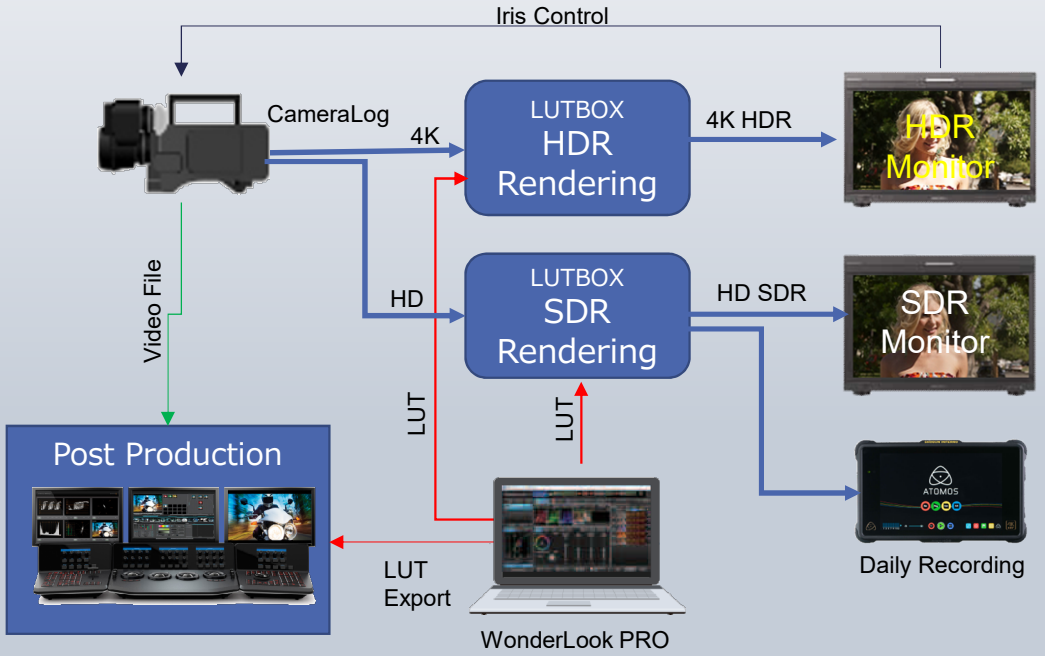


Rec709 with mapping

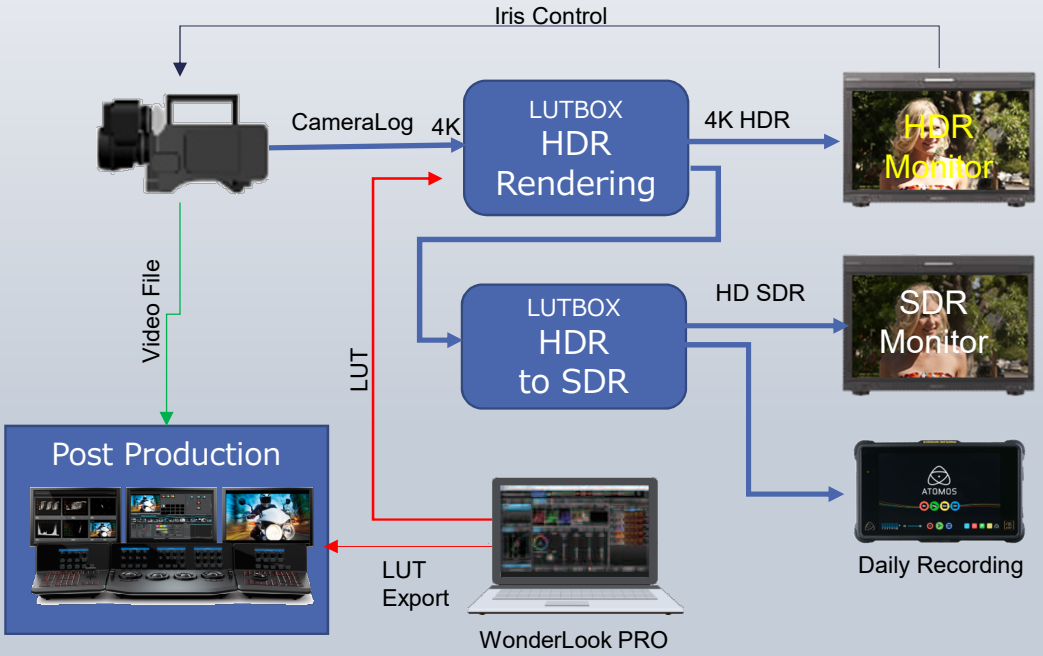


Mappingにより、警告領域が減少し、Rec709色域内に変換されていることが確認できる。

HDR Workflow 1. 映画、TVドラマ

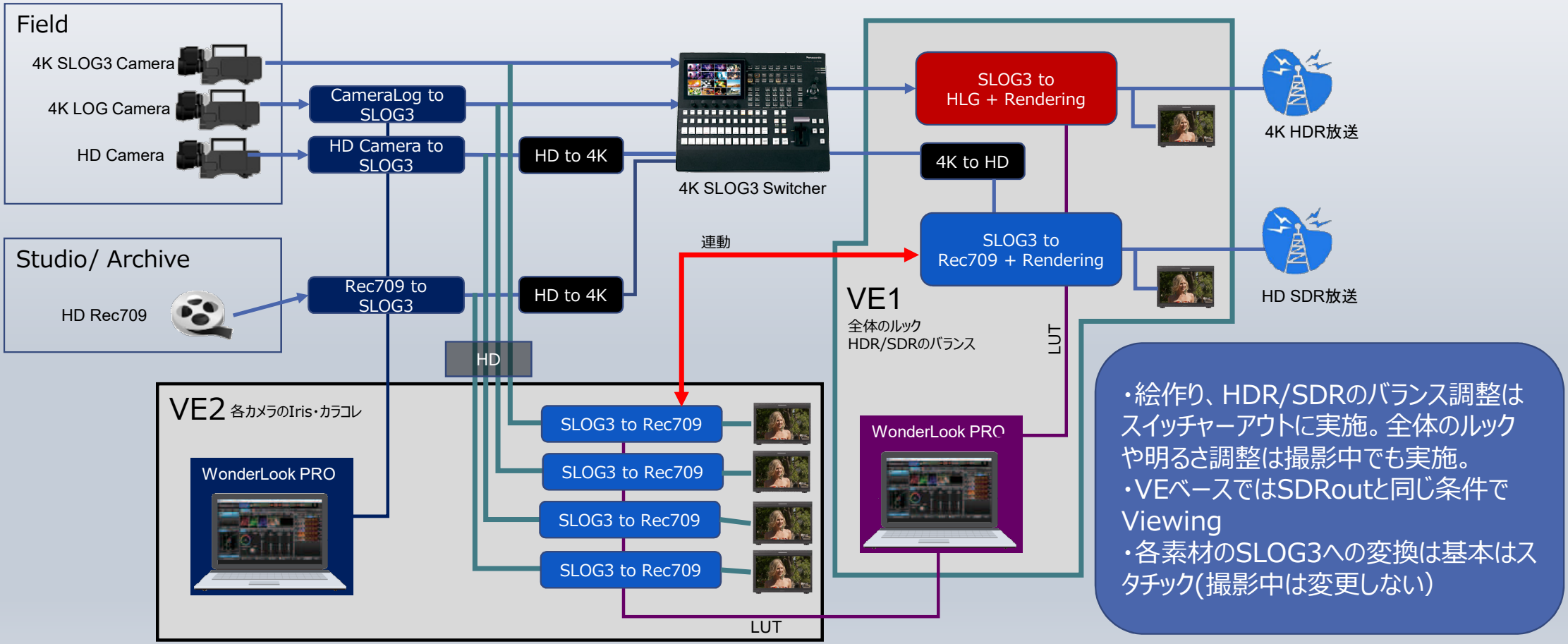


・WLPでIDT(DeLog)+Rendering+HDR/SDRのLUTを作成
・HDR/SDR同時にグレーディング可能
・HDR ViewingでIris決定、SDR outをDailyまたは本編に利用



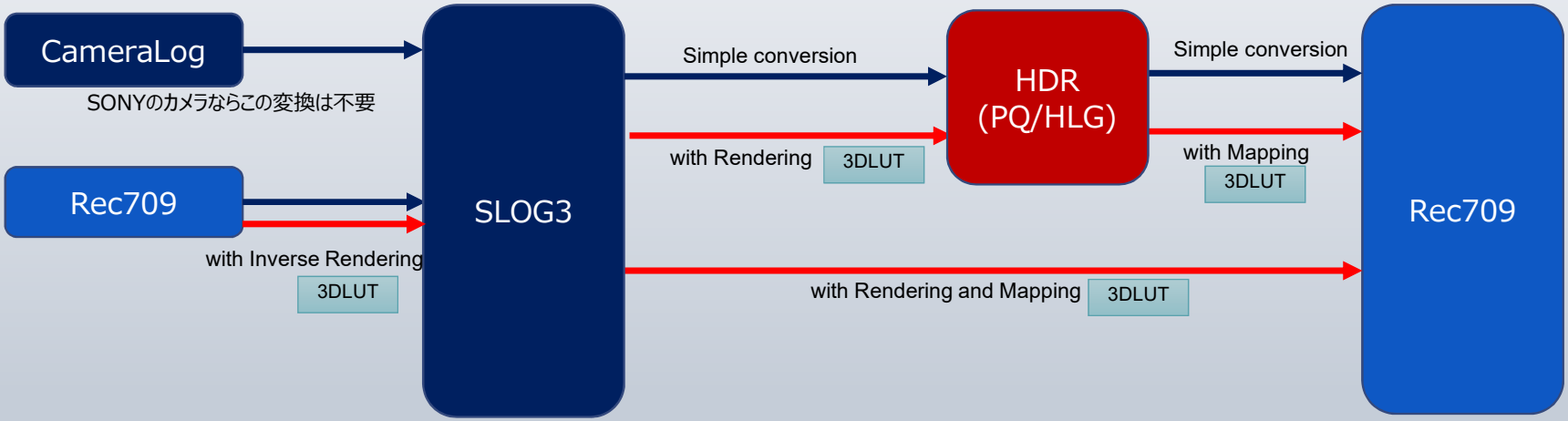
・WLPでIDT(DeLog)+Rendering+HDR/SDRのLUTを作成
・HDR ViewingでGrading,Iris決定
・HDR outをスタチックなHDR to SDRで変換し、Viewing/Daily

HDR Workflow 2. 中継・生放送: SONY提唱のSR Live for HDRをベースに



・絵作り、HDR/SDRのバランス調整はスイッチャーアウトに実施。全体のルックや明るさ調整は撮影中でも実施。
・VEベースではSDRoutと同じ条件でViewing
・各素材のSLOG3への変換は基本はスタチック(撮影中は変更しない)

HDRプロジェクトで必要な各種色変換

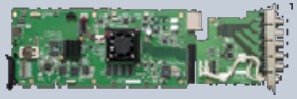


シンプルな色空間変換。計算式/LUTのどちらでも可。

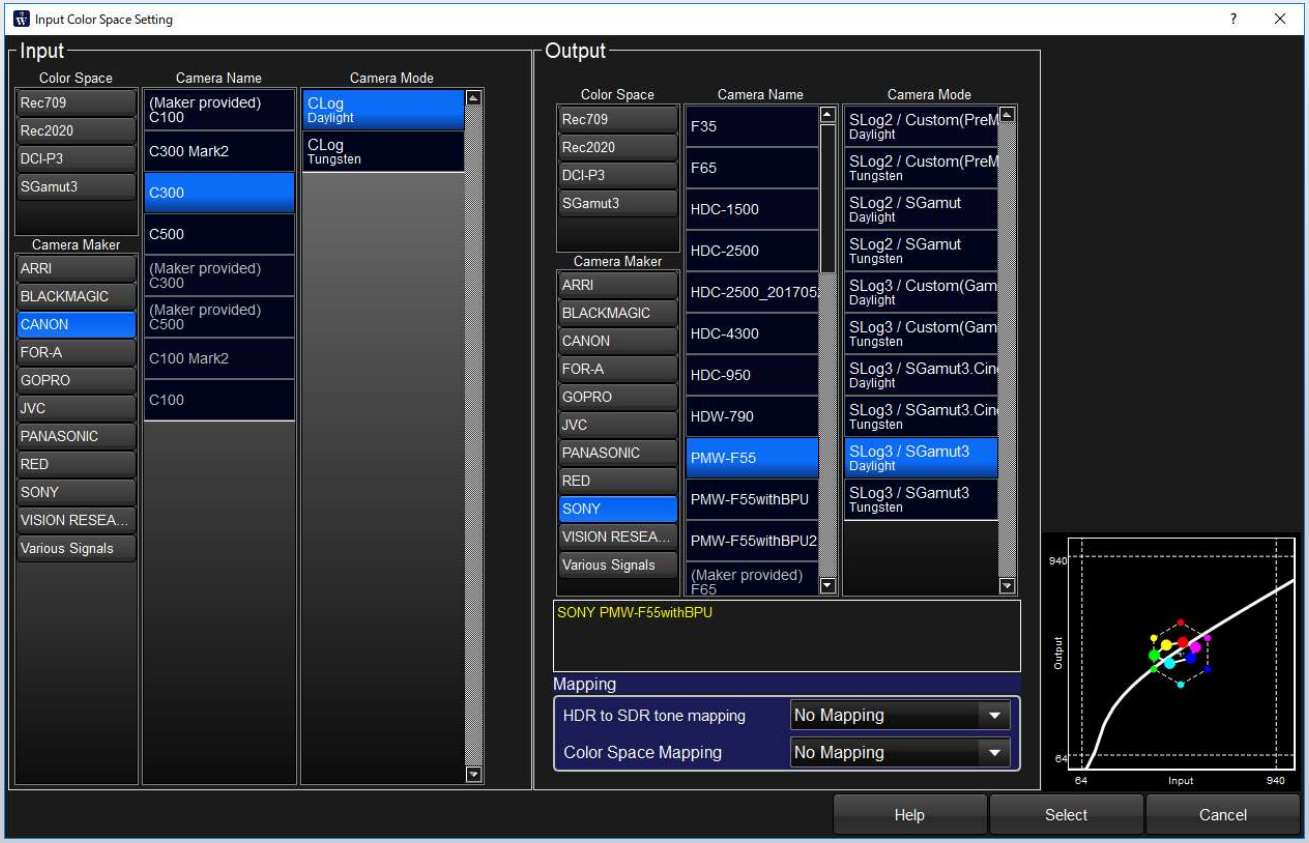
複雑なレンダリングまたはマッピングを含む。LUT処理で実現。

・絵作り(Rendering)不要なプロジェクトはLUTBOX無しでもWF構築は一応可能。
・LUTBOXを活用した方がはるかに柔軟で絵作りの幅が広げられるWFが構築できる。

HDR向け色変換を実現する商品群1. LUTBOX

	対応Format	格子点	ディレイ	付加機能
IS-miniX 	HD/3G-SDI x 1	26^3 三角錐	3μsec スイッチャー 許容範囲内	Frame Capture 静止画再生
IS-miniX Rack4K 	HD/3G-SDI x4	26^3 (x4種) 三角錐	3μsec スイッチャー 許容範囲内	Frame Capture 静止画再生
SB-4024 	HD/3G-SDI x4	32^3 立方体 (7月末に三角錐対応予定)	0.5/1.0frame	4K→HD変換 HD→4K変換 p/i変換
SB-4027	HD/3G-SDI x4 12G-SDI x1	〃	〃	〃
FSI BoxIO	HD-3G-SDI x2	17^3(x2mode時) 立方体	2μsec	WiFi
Teradek COLR	HD-3G-SDI x1	**33**^3 立方体	27μsec	HDMI入力/WiFi

HDR向け色変換を実現する商品群2. WonderLookPro



上記はCanon C-300のClogをSLOG3/SGAMUT3に変換する設定を行っているところ

・GUIで設定した結果がリアルタイムにLUTBOXに送信される。
・HDRに必要な変換はほとんど用意されており、選択するだけで設定は完了する。
・マッピングについてはプリセットとカスタム調整の両方の手段が利用できる。
・HDR撮影支援機能多数搭載

Free Licenseの登録で、ハードウエアがなくても大半の機能を体験できます。色管理やLUTの学習にも役立ちます。
17^3に制限されていますが、Exportも可能であり、DaVinciで種々の変換結果を確認することもできます。

前ページのLUTBOXはすべて制御可能

(おまけ) LUTのその他の活用例1. 色域外警告

SLOG3/SGAMUT3をRRTでレンダリング

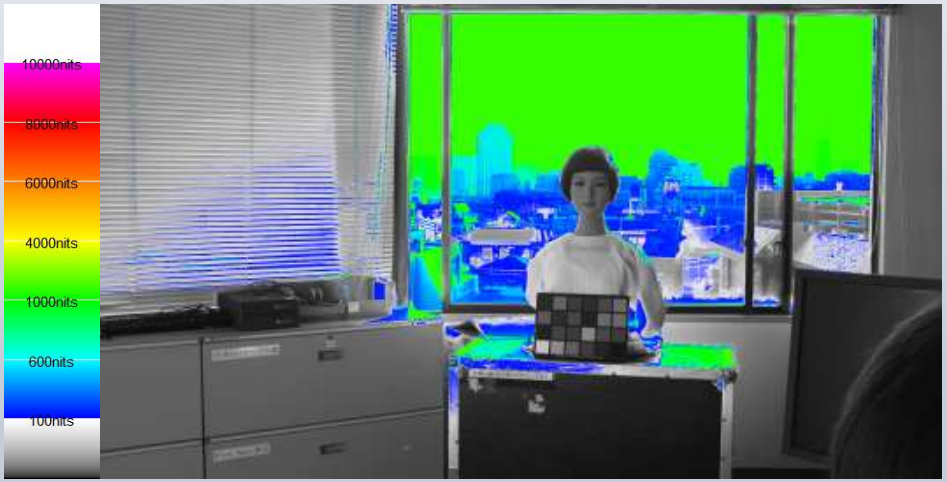
400nits以上の警告

カメラの飽和画素の警告

Rec709色域外の警告

(おまけ) LUTのその他の活用例2. False Color

F-55 SLOG3/SGAMUT3で撮影



被写体輝度域をfalse colorで表示

まとめ

まとめ

課題	どのように理解すべきか
色を変換するとは？	色空間の変換とレンダリングの２種に分類できる
リアルタイムに色変換する方法	計算式とLUT処理の２種の方法がある
LUTの特徴	誤差はあるが、柔軟性、拡張性が高い
LUTの精度に与える要因	格子点数と補間アルゴリズム
マッピングとは？	広い色空間の色を狭い色空間の中に抑えこむこと
HDR/SDRのワークフロー	どこに自由度を持たせるか、によりLUTBOXの活用を決める

・LUTとはどんなものか。精度に与える要因は何か。使用にあたってどのような注意が必要か。
・HDRで必要になる色変換はどのようなものがあるか。LUTはどのように活用すべきか。

本セミナーで紹介した、SB-4024/WonderLookProのソリューションは、6Fにて展示しています。
その他内容についてのお問合せは support@wonderlook.jp までお願いします。
本資料は、2017/7/19以降、http://wowowent.co.jp/is よりダウンロード可能となる予定です。

本セミナーでは扱えなかった項目

項目	概要または私の見解
ファイルフォーマット	.cube,.3dl などグレーディングソフトのテキストファイル形式 .wowlutなどメタデータも含めたバイナリ形式
色変換の非線形性	非線形性が強い変換ほど補間誤差が出やすい。
LUTの格子点数変換	少ない格子点から多い格子点への変換も意味がある場合がある。
LUTの出力側色空間変換	テーブルの数値を再計算して書き換えるだけなので比較的容易
LUTの入力側色空間変換	補間計算が必要なためやや複雑。どうしても必要な場合にのみ実施すべき。
Shaper LUT	1DLUT + 3DLUTの構成にして精度向上を狙う。思わぬ副作用が生じる場合がある。
LUTをどう管理すべきか	難問です。WonderLookProは「管理」が不要となるソリューションを目指します。
LUT管理関連ツール	WLP最新版は管理（変換）機能をかなり強化しています。 Lattice(https://lattice.videovillage.co/)というツールも強力な管理機能を持っているようです。
3DLUTの逆変換について	難題です。もう少し時間をかけて取り組みます。
画像を使った3DLUT作成法	各種ソフトがこの機能を提供しているようです。WLPも実装予定です。
マッピングのより深い考察	ガマットマッピングの考え方、アルゴリズムなど
量子化誤差（ビット数の影響）	10bitでも1DLUTと3DLUTをうまく使えば問題になることはない？