UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una aplicación para gestión de color en entornos multimedia

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Matoses Giménez, Carles

Tutor/a: Igual García, Jorge

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DESARROLLO DE UNA APLICACIÓN PARA GESTIÓN DE COLOR EN ENTORNOS MULTIMEDIA.

Carles Matoses Gimenez

Tutor: Jorge Igual Garcia

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2022-23

València, 30 de septiembre de 2022

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO.ES-CV-000030

Resumen

Este trabajo presenta una introducción a la ciencia del color de forma completa y explica cómo implementarla en software actual, como por ejemplo Blender o DaVinci Resolve. Además, se proporcionan nuevas herramientas para facilitar una correcta implementación de la gestión de color en aplicaciones, como Addons para blender, y facilitar el estudio de las diferencias entre espacios de colores, funciones de transferencia y formatos de imágenes con un nuevo software desarrollado en python.

La aplicación desarrollada ofrece herramientas útiles que no requieren de conocimientos de programación y que muestran de forma gráfica las características de los espacios de color. También incluye facilidades para desarrollar nuevos nodos de forma simplificada con funciones customizadas por el usuario.

Resum

Aquest treball presenta una introducció a la ciència del color de forma completa i explica com implementar-la en programari actual, com ara Blender o DaVinci Resolve. A més, es proporcionen noves eines per facilitar una correcta implementació de la gestió del color en aplicacions, com Addons per a Blender, i per a facilitar l'estudi de les diferències entre espais de colors, funcions de transferència i formats d'imatges amb un programari desenvolupat en Python.

L'aplicació desenvolupada ofereix eines útils que no requereixen coneixements de programació i que mostren de forma gràfica les característiques dels espais de color. També inclou facilitats per desenvolupar nous nodes de forma simplificada amb funcions personalitzades per l'usuari.

Abstract

This work presents a comprehensive introduction to the science of color and explains how to implement it in current software, such as Blender or DaVinci Resolve. Additionally, new tools are provided to facilitate proper color management implementation in applications, like Addons for Blender, and to facilitate the study of differences between color spaces, transfer functions, and image formats using Python-developed software.

The developed application offers useful tools that do not require programming knowledge and visually display the characteristics of color spaces. It also includes facilities to simplify the development of new nodes with user-customized functions.

Este trabajo ha sido posible gracias a muchas personas con pasión e interés por el mundo del procesado de imágenes y el color. En primer lugar, quiero expresar mi agradecimiento a mi tutor Jorge Igual Garcia por ofrecerme orientación en algunos ejemplos y cálculos empleados en el trabajo. También, quiero agradecer a la universidad por abrirme las puertas a la programación y manipulación de imágenes a un nivel profesional.

No puedo dejar de reconocer a todos los entusiastas del color digital que generosamente comparten sus proyectos y conocimientos. Su contribución desinteresada enriquece nuestra comunidad y ha sido una fuente inestimable de inspiración.

Por último, quiero agradecer el apoyo y paciencia de mi familia que soportaron mis explicaciones y presentaciones sin sentido durante todos estos meses.

Índice general

I Memoria

Índice de figuras

Índice de tablas

Listado de siglas empleadas

ACES Academy Color Encoding System.

CGI Computer Generated Image.

CIE Commission Internationale de l'Eclairage.

DN Digital Number.

DR Dynamic Range.

DRT Display Render Transform.

LMT Ocio look up table en nomenclatura Aces.

OCIO Open Color IO.

PCS Profile Connection Space.

Parte I

Memoria

Motivación del documento

El objetivo de este documento es familiarizar a artistas y generalistas con la terminología empleada en la gestión de color y mostrar los pasos de forma ordenada para hacer un uso correcto de las operaciones necesarias para transmitir y almacenar contenido digital. Para ello se ofrecen herramientas creadas y explicadas en este trabajo que facilitan la implementación de la gestión de color y el estudio de las propiedades de los espacios de colores, funciones de transformación, matrices de transformación y características de espacios de color para el renderizado.

El contenido gratuito que se puede encontrar para estudiar de forma autodidacta suele estar mal explicada y se centra únicamente en una parte del procesado de imágenes. Esto no permite hacerse una idea completa de la correcta gestión de la información o incluso conlleva a malentendidos y confusiones que luego se pueden difundir a nuevos usuarios. Con este resumen de la manipulación de una imagen desde que es capturada hasta que es mostrada al destinatario final, se rellena este vació actual de una guía gratuita y concisa de gestión de color para artistas.

Tras trabajar en algunos proyectos profesionales y en múltiples cortos para la universidad, he conseguido experiencia suficiente para explicar los fundamentos de la imagen digital a nuevos usuarios. Mi función en equipos de generación espontánea en la universidad a sido de supervisor en efectos visuales, composición, CGI y gradación de color. La necesidad de explicar todos los conceptos necesarios de estas áreas me ha llevado a generar una guía de gestión de color específica para usuarios de aplicaciones de edición de vídeo y diseño 3D.

Las aplicaciones emplean sus propios sistemas de gestión de color. Es importante entender los conceptos generales para aplicarlos a las necesidades de cada proyecto dentro del funcionamiento de cada programa. Es muy común encontrar errores o descripciones dentro de algunos software que confunden a los usuarios. Por esta razón, el programa realizado en este documento se puede emplear para confirmar que el comportamiento de las aplicaciones es el deseado y que no realizan operaciones que pueden estropear el procesado de una imagen.

Por último, es importante elegir las herramientas correctas en un flujo de trabajo para no desperdiciar espacio de disco duro o potencia de procesado. Cada archivo tiene sus funciones y especificaciones para usos concretos. Dado que el objetivo de este trabajo es proporcionar facilidades para usuarios con poca experiencia en aspectos técnicos, se proporcionan herramientas para leer y estudiar imágenes desde varias extensiones.

Capítulo 1

Ciencia del color digital

La ciencia del color digital se refiere al **estudio y la comprensión de cómo se crean, miden, capturan, reproducen y perciben los colores en los sistemas digitales** [1]. Esto abarca muchos mas campos además de la gestión del color los cuales no suelen ser mencionados al hablar del procesado de imágenes. A continuación se explorarán todos los procesos, desde la captura de la luz a su muestra por una pantalla pasando por compresiones y transformaciones.

1.1. Paradigma actual

La ciencia del color digital **es extensa y abarca un gran grupo de disciplinas**. En primer lugar, el desarrollo de este campo ha ocurrido de forma muy rápida y **en función de las necesidades de un hardware siempre en evolución**.

Estas circunstancias han dado lugar a una nomenclatura y variedad de soluciones a problemas específicos muy amplia que genera una barrera para nuevos usuarios. Para ejemplificar este problema sin agregar una dificultad innecesaria se expondrán solo algunos casos comunes que se pueden observar día a día. Todos los conceptos se definirán y mostrarán posteriormente en este documento.

Las pantallas de los dispositivos móviles, pantallas de ordenador y televisores, en casos habituales, pueden emitir la misma cantidad de colores, a este grupo de colores se le llama sRGB. Sin embargo, al observar el mismo contenido desde diferentes dispositivos se observan algunas diferencias. Esto es debido a la curva opto-electric transfer function, esta curva transforma un espacio de color con sus primarios a una imagen más agradable a los ojos humanos. Las televisiones emplean una curva llamada rec.709 mientras que las pantallas de ordenador o de móvil suelen emplear una curva llamada sRGB (mismo nombre que el espacio de color). Para complicar más las cosas, existen dos curvas llamadas sRGB, una de ellas es descrita como “gamma 2.2” mientras que otra es una aproximación y suele ser representada como “$\sim$gamma 2.2” y que difiere sobretodo en las sombras.

Los espacios de colores pueden tener más características aparte del rango de colores que generarán su aspecto final y es necesario entenderlas para una correcta gestión y tratamiento de color.

1.2. Scene Refered y Display Referred

Los conceptos **Scene Refered** y **Display Referred** provienen de la necesidad de distinguir las **escenas reales donde se captan las fotografías** (campo de flores a plena luz del día) y los **dispositivos donde se reproducen las imágenes procesadas** (pantallas, televisiones y proyectores).

Scene Referred: Son imágenes que representan la luz de forma lineal tal y como fue obtenida de la escena real.

Display Referred: Son aquellas imágenes preparadas para ser mostrada por pantalla. Al igual que hacen los ojos, los valores de luz lineales se les aplica una curva logarítmica para simular que la luz se ve con ojos humanos 1.1. Este proceso también se aplica para conservar una mayor cantidad de información al almacenar las imágenes en un archivo.

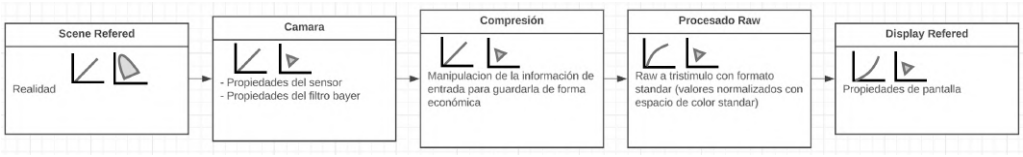


Figura 1.1: Proceso de una escena real a una pantalla por medio de una cámara en formato RAW.

1.3. Conceptos básicos de la luz

Para entender conceptos como rango dinámico y primarios, es necesario empezar por el comportamiento real de la luz y su naturaleza.

1.3.1. Unidades

La magnitud empleada para la luz varia mucho dependiendo de su uso, este documento se centrará en los empleados en el transcurso de la vida de una imagen digital:

- **Fotones** :
  - Se trata de la propia luz y de su frecuencia (color).
  - Los sensores reciben los fotones.

- **Candela**:
  - Unidad básica del Sistema Internacional que mide la **intensidad luminosa**.
- **Nits**:
  - Un nit equivale a una candela por metro cuadrado.
  - Se empelará esta unidad para hablar de la luz emitida por las pantallas y otros dispositivos.
  - Normalmente se emplean tecnologías con 100 nits en dispositivos normales y 1000, 2000, 4000 o 10000 nits en las pantallas HDR.
- **Watts**:
  - 0.05W = 1Candle

- Unidad empleada en renderizados 3D y simulaciones de trazado de rayos
- Como referencia [2]:

  - Cielo despejado = $1000W/m^2$
  - Cielo nublado = $500W/m^2$
  - Cielo muy nublado = $200W/m^2$
  - Noche con luna llena = $0{,}001W/m^2$

1.3.2. Traducción de colores reales a digitales

Por norma general, no se emite una única longitud de onda como puede ser la 510nm para la luz verde, sino un **espectro**, **múltiples frecuencias combinadas** 1.3, 1.4. **Los objetos sobre los que incide la luz reflejan un nuevo espectro**, en este caso alterado por las propiedades del material (se observan las frecuencias de luz no absorbidas por el objeto). De este mismo modo, **el espectro de la luz emitida también tendrá un efecto directo en la percepción del color** resultante. Un claro ejemplo de este fenómeno serian las luces ultravioletas que emplean en las discotecas para hacer brillar ciertos colores y no afectar a otros. Este tipo de luz genera fluorescencia al reflejarse sobre ciertos materiales.

Para entender como pasar de un espectro de luz a componentes digitales representables se emplea el concepto de **metamerismo**. El **metamerismo** es un fenómeno en la cual dos muestras de color coinciden bajo unas condiciones determinadas (fuente de luz, observador, geometría...) pero no bajo otras diferentes. Para transformar los **espectros reales** en valores representables por pantallas, es necesario pasarlos **a los componentes RGB** que emiten las leds monocromáticas y generar un **nuevo espectro con la misma respuesta visual**.

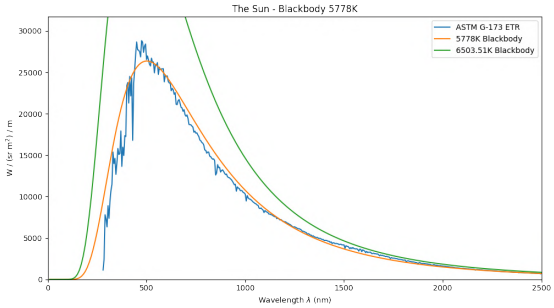


Figura 1.2: Espectro de luz D65 y D57

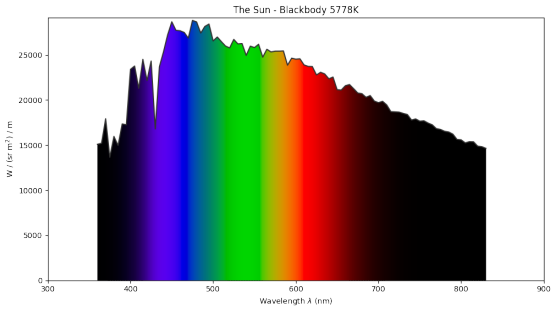


Figura 1.3: Espectro de luz D57

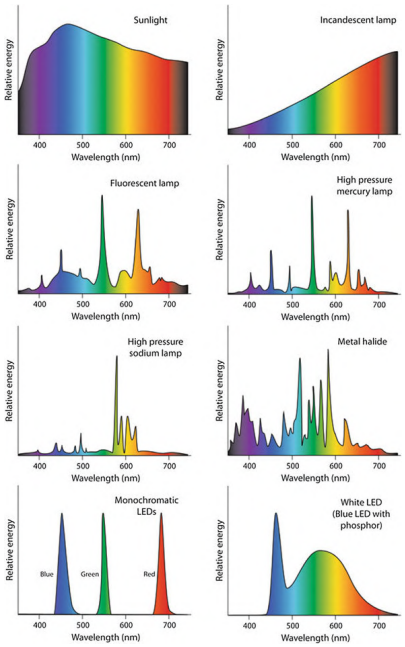


Figura 1.4: Diferentes espectros de fuentes de luz [3]

Basada en la **premisa de que cualquier color puede ser descrito como una combinación de tres colores primarios** (rojo, verde y azul) se declaro el **espacio de color CIE XYZ** [4] que contiene **todos los colores visibles por el ojo humano** . Fue establecido a partir de muchos experimentos realizados por la Comisión Internacional de la Iluminación (CIE). William David Wright y John Guild llevaron a cabo una serie de experimentos para medir la sensibilidad de los ojos humanos a la luz monocromática en diferentes longitudes de onda. **Los valores XYZ** oscilan entre 0-100 unidades y cada coordenada **representa la cantidad de energía que se percibe como rojo, verde y azul**. La coordenada Y representa la luminancia o brillo del color, la coordenada X representa la cantidad de energía de luz que se percibe como roja-verde y la coordenada Z representa la cantidad de energía de luz que se percibe como azul-amarillo.

En la imagen 1.5 se muestra como el espectro del sol pasa del valor tristímulo $X = 95{,}047; Y = 100; Z = 108{,}883$, a los valores $x = 0{,}3128; y = 0{,}3290$ para representarlos en dos dimensiones. Los valores XYZ se emplearán como el valor intermedio para pasar del espacio XYZ a xyY del siguiente modo:

1. Calcular el valor de la luminancia Y del color. Este valor se encuentra en la coordenada Y de las coordenadas XYZ.
$$Y = Y$$
2. Calcular los valores de x e y dividiendo las coordenadas X y Z por la suma de las coordenadas X, Y y Z:
$$x = \frac{X}{X + Y + Z}$$
$$y = \frac{Y}{X + Y + Z}$$

3. Normalizar los valores de x e y dividiéndolos por el valor de luminancia Y.

$$x' = \frac{x}{Y}$$
$$y' = \frac{y}{Y}$$

4. Las coordenadas cromáticas xy del color son los valores normalizados x' e y', respectivamente.

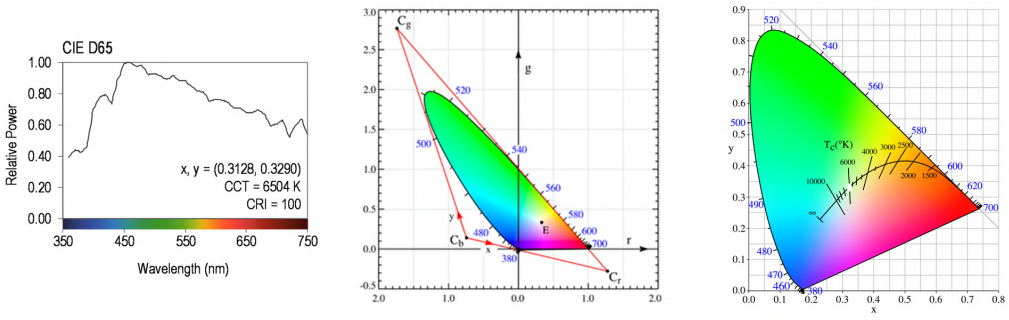


Figura 1.5: Transición de un espectro a XYZ

La transformación de colores XYZ a diferentes espacios de color RGB deben ser perceptibles de forma similar siempre que estén mostradas en una pantalla con el espacio de color correspondiente. Al transformar XYZ 1.6 a otro espacio, los valores se reposicionan de forma que presenten una respuesta similar al valor percibido en una escena real. **En el espacio sRGB 1.8, los parches se encuentran más cerca del centro mientras que en el espacio ACEScg 1.7, los parches se muestran en la misma posición que las coordenadas xy reales** mostradas en la tabla 1.1.

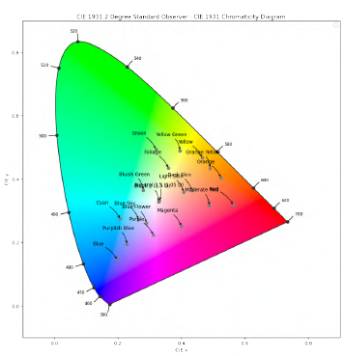


Figura 1.6: Espectro visible. Colores reales de los parches N Ohta representados en xy

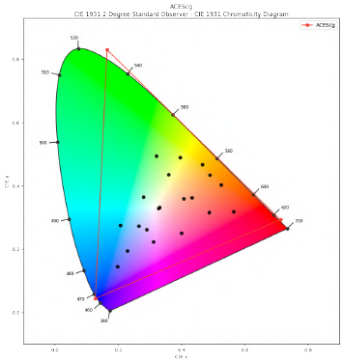


Figura 1.7: ACEScg. Mantiene el color real

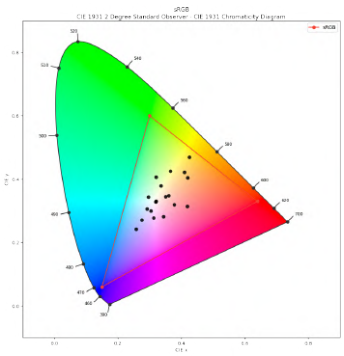


Figura 1.8: sRGB. Se comprimen los colores para ser representables.

Nombre	Valor XYZ
black 2 (1.5 D)	`[ 3.18668376, 3.35489375, 3.81660566 ]`
blue	`[ 8.41232639, 6.23027902, 30.00714132 ]`
blue flower	`[ 25.83217459, 24.38130492, 45.33530386 ]`
blue sky	`[ 17.85793538, 19.0802859, 34.544716 ]`
bluish green	`[ 31.2790205, 42.72969745, 44.71402208 ]`
cyan	`[ 14.47677825, 19.86681263, 39.53573603 ]`
dark skin	`[ 10.97085572, 9.70278591, 6.05562778 ]`
foliage	`[ 10.1081878, 12.98479141, 6.69391722 ]`
green	`[ 14.50133246, 23.57046102, 9.52096407 ]`
light skin	`[ 38.13368873, 35.58313054, 25.94134675 ]`
magenta	`[ 29.41768334, 19.26873993, 30.28879001 ]`
moderate red	`[ 28.45945838, 19.22703363, 13.75429023 ]`

Tabla 1.1: Ejemplos de coordenadas XYZ de parches de color extraídos de N Ohta

Este proceso afecta a la percepción de los colores en diferentes dispositivos de salida ya que **permite que una pantalla sRGB que no puede representar todos los colores, pueda mostrar diferencias entre colores fuera de sus limites RGB**. En este caso, un parche verde se verá menos saturado en una pantalla sRGB que en una pantalla ACEScg (no existen pantallas que puedan mostrar el rango de primarios ACES). Por ende hay que intentar que la percepción de un parche del que conocemos el color real se vea similar en múltiples dispositivos. Este proceso no es simple y suelen añadirse procesos artísticos para generar un color “más verosímil o placentero”.

1.4. Imagen Raw

El procesado raw es un buen ejemplo para entender la gestión de color, por esta razón se explicará paso a paso el proceso para introducir los conceptos con un ejemplo real.

En las **cámaras analógicas**, de forma muy simplificada, se registra la luz que entra por la lente en una **película fotográfica (material fotosensible)**. Durante el tiempo de exposición, la membrana cambia su color a negro (invertido luego en el revelado).

Las **cámaras digitales** sustituyen este elemento (película fotográfica) por un **sensor compuesto de píxeles**. En ellos se almacena el valor de fotones recibidos durante la exposición de la fotografía 1.9. El Rango Dinámico esta limitado por el sensor (cantidad máxima y mínima de fotones que puede detectar) y los niveles de saturación y ruido de los píxeles. Con un DR limitado, el trabajo de la cámara o el usuario es elegir que porción de la luz conservar.

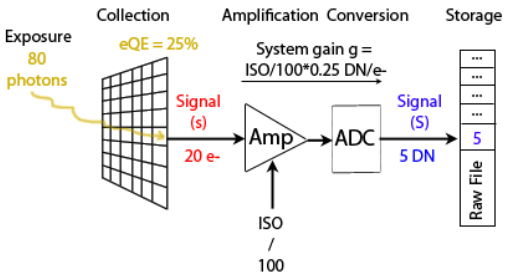


Figura 1.9: Proceso de un sensor de fotones a valores digitales (DN)

Los formatos de imagen que pueden almacenar este tipo de datos se llaman formatos raw:

- Nikon: NEF
- Canon: CR2
- Sony: SR2
- Pentax: PTX
- Olympus: ORF
- Fujifilm: RAF
- Panasonic: RAW2

1.4.1. De RAW a Imagen

El archivo raw contiene una matriz de 2 dimensiones con los valores RGB en patrón bayer 1.10 y Scene Referred. La cantidad de píxeles de la imagen es la misma que la cantidad de píxeles del sensor.

$$\begin{bmatrix} R & G & R & G & R & G \\ G & B & G & B & G & B \\ R & G & R & G & R & G \\ G & B & G & B & G & B \end{bmatrix} \text{ Donde la unidad mas pequeña es } = \begin{bmatrix} R & G \\ G & B \end{bmatrix}$$

Figura 1.10: Patron Bayer

El sensor de la cámara esta recubierto por un filtro que deja pasar determinadas frecuencias a determinados píxeles y cada pixel solo almacenará un único color. Más tarde serán interpretados como Rojo Verde y Azul. El patrón del filtro depende del distribuidor y estará documentado en los metadatos raw.

Los pasos seguidos para procesar la imagen se pueden separar en los siguientes pasos.

1. **Eliminar los niveles de negro de cada canal**. Los canales RGB contienen la cantidad de DN (digital number) generada como respuesta a los fotones a los que fue expuesto el sensor.

En el caso de carecer de una entrada de luz, los píxeles del sensor registran un mínimo de energía producida por la cámara (este valor depende del modelo de la cámara) y debe ser substraído. Cada cámara y proveedor eligen los valores dependiendo del hardware.

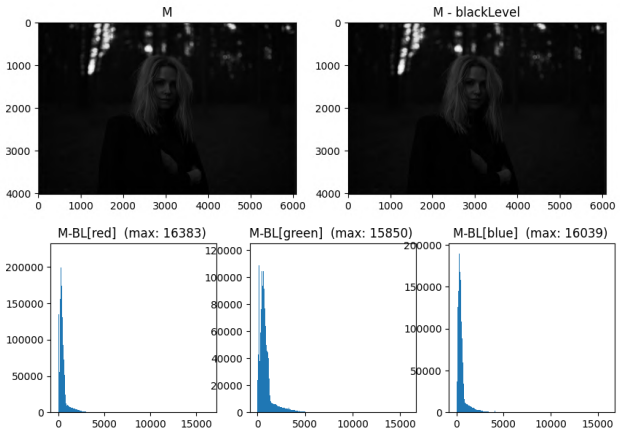


Figura 1.11: Matriz Bayer - [0, 0, 0, 0]

En este ejemplo, los metadatos almacenan un Black Level de $BL_R = 0; BL_G = 0; BL_G = 0; BL_B = 0;$

2. **Balance de blancos**. Para que los colores se perciban de forma natural, se recrea el efecto de adaptación cromática y una compensación de la forma de capturar los colores por el sensor. El balance de blancos se traduce como la multiplicación de un valor por cada canal para conseguir que un color determinado se perciba como blanco puro. En los archivos raw, se puede almacenar como un metadato más. En este caso, por los metadatos, el balance de blancos calibrado en la escena es de [1.74, 1, 1.58, 1] correspondiente a los canales $[R, G, B, G]$

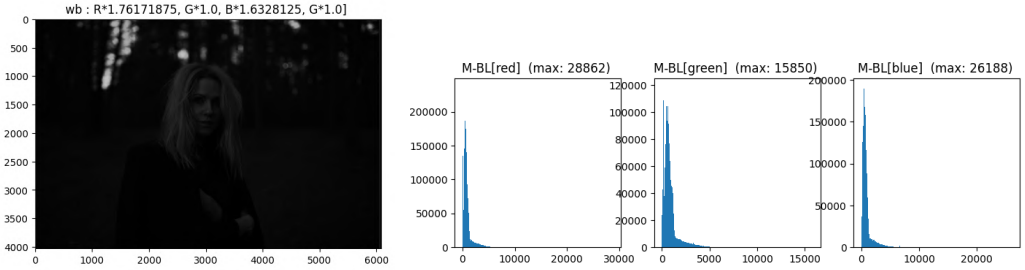


Figura 1.12: Balance de blancos

3. **Corregir nivel de luz** (opcional, solo en uint16).
Tras realizar la fotografia y acceder a los píxeles del archivo, la luz (almacenada de forma lineal) contiene el rango (que depende de la camara) de 14 EV. La imagen esta codificada en 14 bits y el sensor puede registrar estos 14 pasos de luz evitando almacenar ruido.

   En el caso de estar editando esta imagen en un entorno de 16 bits uint, al representar las imágenes con el valor mínimo=0 y el valor máximo=2**16-1, la imagen se observaría 4 veces más oscura de lo que se apreciaría en la fotografía original. La solución en ese caso seria multiplicar la imagen por 4 (2 pasos de luz) y de este modo repartir de forma lineal los

valores de 14 bits en 16 bits. En el caso que nos ocupa, estamos empleando un entorno de float32 (generalmente empleado por todos los programas de edición) y para representar la imagen la normalizamos a partir del mayor numero comprendido en la matriz, por lo que no es necesario remuestrear los valores.

Para demostrar este comportamiento, en la figura 1.13 se muestra como dependiendo del valor máximo representable, la imagen es más luminosa o menos a cambio de perder detalles en las partes más brillantes.

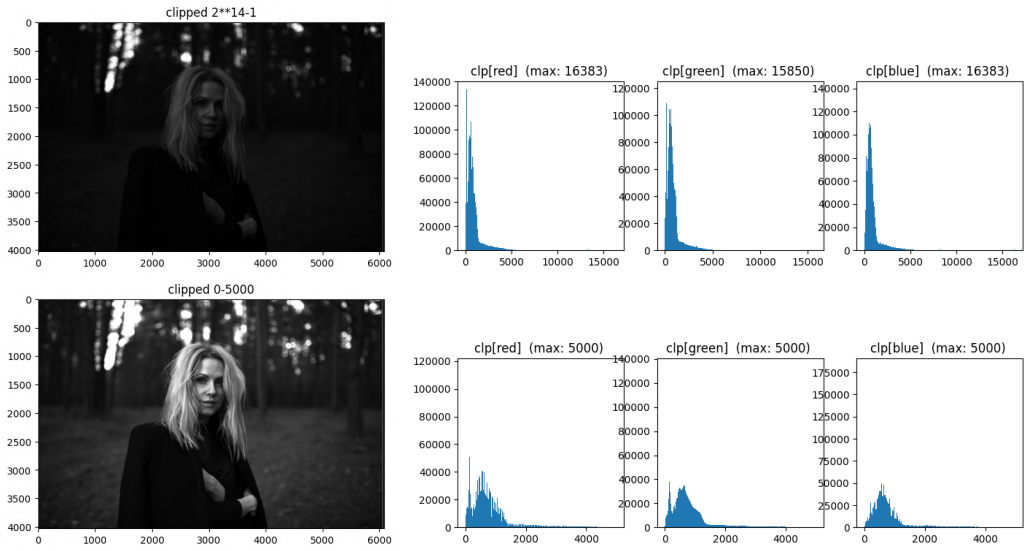


Figura 1.13: máximo punto de luz

4. **Reconstrucción de altas luces**.
   Tras el balance de blancos (multiplicación de un valor por cada canal), el nivel máximo de los canales rojo y azul suele sobrepasar el valor máximo del canal verde. Si representamos la imagen normalizada de nuevo se produce un efecto no deseado 1.14.

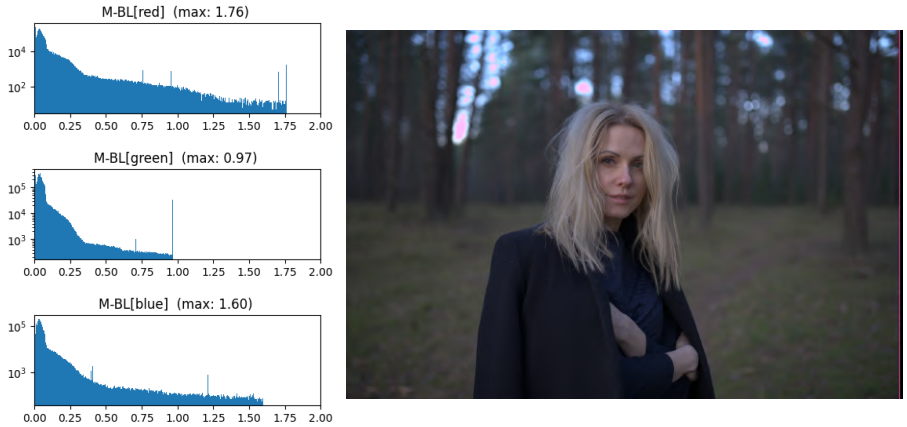


Figura 1.14: Histograma del Balance de blancos

El canal verde no conserva información más allá de los 14 EVs de rango dinámico con los

que se tomo la fotografía.

Para solucionarlo se puede emplear diferentes algoritmos de reconstrucción para las altas luces. El objetivo es recuperar un rango dinámico más amplio que el obtenido simplemente eliminando la parte sobrante de los 14 EVs.

- **Clip**:
  Perdemos las altas luces y casi 2/3 de un EV 1.15.

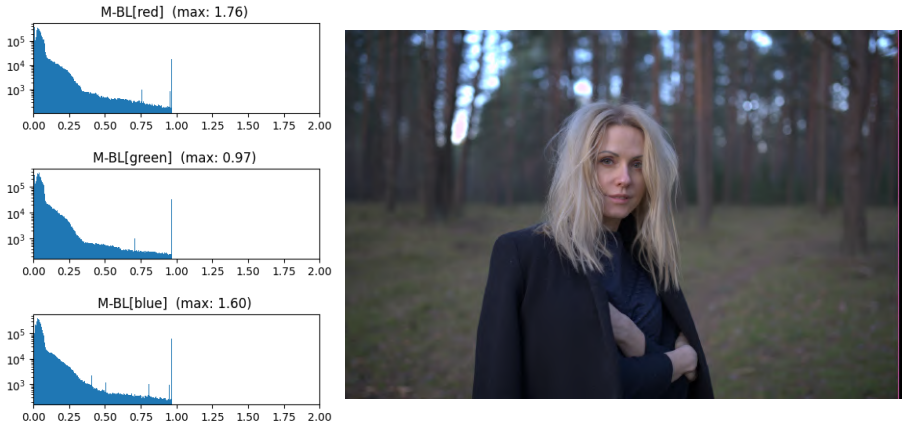


Figura 1.15: Clipped highlights

- **Mean**:
  Recuperamos ese 2/3 de EV adicional 1.16. La imagen parece más oscura pero es importante recordar que esto se debe a que estamos representando todo el rango dinámico (normalizando la imagen).

$$pixel_{Gi} = \begin{cases} (pixel_{Ri} + pixel_{Bi})/2 & \text{if} pixel_{Gi} > umbral \\ pixel_{Gi} & \text{else} \end{cases} \tag{1.1}$$

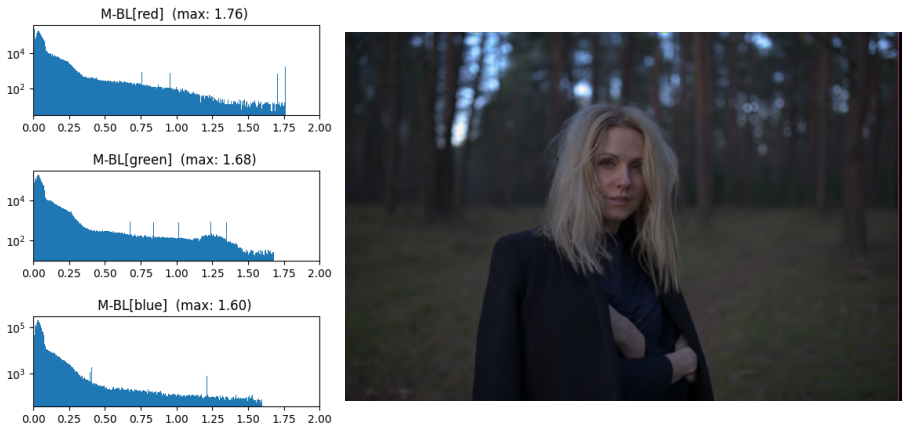


Figura 1.16: Mean highlights

Hay otros metodos para recuperar altas luces, estos son algunos empleados por el software darktable:

- Reconstruir en LCh: Analizar cada píxel con al menos un canal recortado e intentar corregir el píxel recortado (en el espacio de color LCh) utilizando los valores de los otros píxeles (3 para Bayer o 8 para X-Trans) en el bloque del sensor afectado.
- Reconstruir color: Utilizar un algoritmo que transfiera información de color desde las áreas sin recortes hacia los resaltes recortados.
- Laplacianos guiados: Utilizar un algoritmo (derivado del módulo de difusión o de enfoque) para replicar detalles de canales válidos en canales recortados y propagar gradientes de color desde regiones válidas circundantes hacia regiones recortadas.

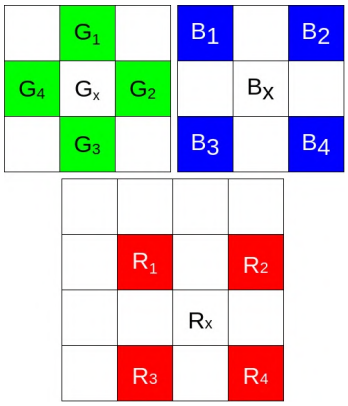
5. **Demosaico**:
   Durante este proceso, la matriz es todavía de un único canal con el formato bayer 1.10 que corresponda. Las operaciones se han realizado afectando a los píxeles correspondientes a los futuros canales.

   En el paso de demosaico se combinarán los píxeles adyacentes para generar los componentes que faltan 1.17. El canal original de cada píxel no tiene por que conserva el mismo valor tras el demosaico.

$$\begin{bmatrix} R & G & R & G \\ G & B & G & B \\ R & G & R & G \end{bmatrix} \text{ Demosaico} = \begin{bmatrix} Rgb & rGb & Rgb & rGb \\ rGb & rgB & rGb & rgB \\ Rgb & rGb & Rgb & rGb \end{bmatrix}$$

Figura 1.17: Demosaico.

- **demosaicing CFA Bayer bilinear**
  El proceso de interpolación lineal consiste en tomar los valores de color de los píxeles adyacentes y calcular un valor promedio ponderado para el píxel faltante. Por ejemplo, para un píxel verde faltante, se tomaría el promedio de los valores de color verde de los píxeles verdes adyacentes.



$$R_x = \frac{1}{4}x(R1 + R2 + R3 + R4)$$
$$B_x = \frac{1}{4}x(B1 + B2 + B3 + B4)$$
$$G_x = \frac{1}{4}x(G1 + G2 + G3 + G4)$$

Figura 1.18: Demosaico Bilineal donde R_x B_x y G_x son los píxeles que no poseen el correspondiente canal

- Otros algoritmos de demosaico podrían ser:
  - demosaicing CFA Bayer Malvar2004
  - demosaicing CFA Bayer Menon2007

- demosaicing CFA Bayer DDFAPD

6. **Transformación de color y corrección**

Cada componente rojo, verde y azul que han sido obtenidas, solo tienen sentido si se entiende como se traducen estos valores al mundo real. El sensor ha percibido los colores de forma diferente a los ojos humanos. Para determinar que la percepción de rojo es la misma que la de la cámara o en su defecto, para corregirla, se hacen muchas pruebas con parches de colores de los que ya se conoce la respuesta del ojo humano (un ejemplo de estos parches son los N Ohta mencionados en la tabla 1.1).

a) El primer paso seria conseguir estos parches y ese proceso no es fácil. Para manufacturar un parche que emita una respuesta de la que se conozca su resultado requiere de mucha precisión y colores de impresión especiales 1.61.1.

Figura 1.19: Representación sRGB de unos parches de color

b) Tras exponer la cámara a los parches, se compara los resultados y se genera una matriz con vectores que desplazan cada color a la posición XYZ del parche correspondiente. Esta matriz nos indica como pasar de el espacio XYZ a el espacio nativo de la cámara y viceversa.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Por suerte, el fabricante de la cámara ya ha realizado este trabajo. Esta matriz también se encuentra en los metadatos de los archivos raw como M^{-1}.

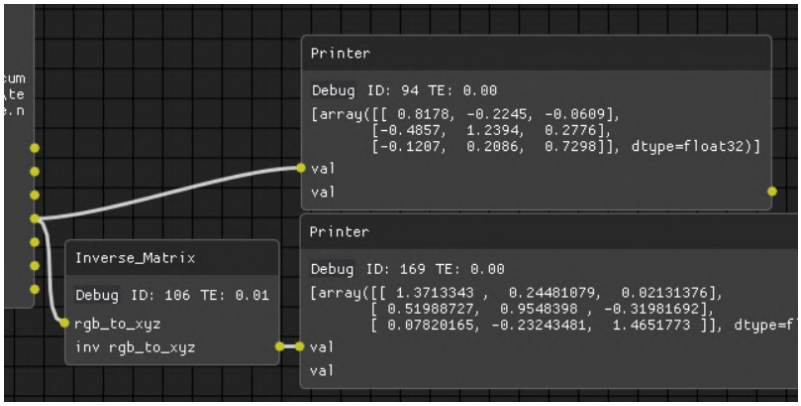


Figura 1.20: Matriz de los metadatos y matriz inversa.

Pero hay un problema con esta M: un punto blanco $R = G = B = 1$ debe producir una luminancia $Y = 1$.
Esta M no lo cumple. Así que hay que normalizar los valores para que se muestre un punto blanco en la posición de interés del usuario.

$$
\begin{aligned}
X &= 1{,}3713 \cdot \alpha \cdot R + 0{,}2448 \cdot \beta \cdot G + 0{,}0213 \cdot \gamma \cdot B \\
Y &= 0{,}5198 \cdot \alpha \cdot R + 0{,}9548 \cdot \beta \cdot G - 0{,}3198 \cdot \gamma \cdot B \\
Z &= 0{,}0782 \cdot \alpha \cdot R - 0{,}2324 \cdot \beta \cdot G + 1{,}4651 \cdot \gamma \cdot B
\end{aligned}
$$

Figura 1.21: Matriz con Punto blanco

¿Cómo calcular α, β, γ? Se especifica un iluminante (blanco) dado y se resuelve el sistema. Por ejemplo:

$$
\begin{aligned}
&\text{Iluminante D65: } X = 0{,}9505, Y = 1{,}0000, Z = 1{,}0891 \\
&\text{Iluminante D50: } X = 0{,}9642, Y = 1{,}0000, Z = 0{,}8252
\end{aligned}
$$

Para el caso del D65, sustituyendo 1.22:

$$
\begin{aligned}
0{,}9505 &= 1{,}3713 \cdot \alpha + 0{,}2448 \cdot \beta + 0{,}0213 \cdot \gamma \\
1 &= 0{,}5198 \cdot \alpha + 0{,}9548 \cdot \beta - 0{,}3198 \cdot \gamma \\
1{,}0891 &= 0{,}0782 \cdot \alpha - 0{,}2324 \cdot \beta + 1{,}4651 \cdot \gamma
\end{aligned}
$$

Figura 1.22: Sistema de ecuaciones para obtener α, β, γ para D65.

Se resuelve el sistema. Se obtienen los coeficientes y la nueva transformación 1.23:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}6670 & 0{,}2644 & 0{,}0189 \\ 0{,}2529 & 1{,}0313 & -0{,}2842 \\ 0{,}0380 & -0{,}2510 & 1{,}3020 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}
$$

Figura 1.23: Nueva matriz con punto blanco D65 multiplicada por cada píxel RGB

Esta matriz cambia según la cámara y el iluminante considerado.

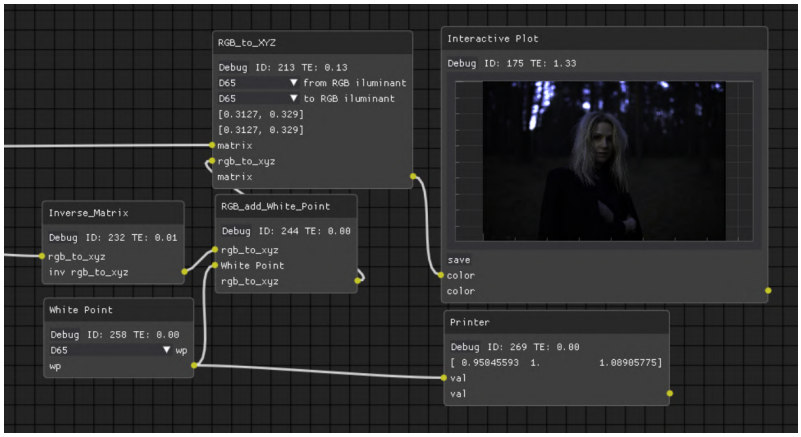


Figura 1.24: Imagen en CIE XYZ con iluminante D65. En la imagen se muestra el proceso para convertir de una matriz M^{-1} a M con $D65$

c) Con la matriz en un espacio de color estandarizado, se puede transformar a cualquier otro espacio de color. En este caso caso se muestra en sRGB en D65 y en D50.

Figura 1.25: sRGB lineal D65

Figura 1.26: sRGB lineal D50

Cuando la fotografía este transformada a un espacio de color con iluminante D65, el color percibido como "blanco" será el mismo que el color emitido por el espectro D65 que equivale a un día soleado. La elección de un iluminante suele estar relacionada con la iluminación del entorno donde se expondrá el contenido.

7. **Apply Gamma**
   Posteriormente se profundizará en la importancia real de la función gamma y OETF (opto-electric-transfer-function). Por el momento solo se destacará que el espacio sRGB utiliza un $gamma = pixel^{1/2,2}$ y una pantalla de espacio sRGB emplea un $gamma = pixel^{2,2}$ 1.27. Por tanto, la señal original que oscila entre valores 0 (negro) y 1 (blanco), si no se ha escogido rescatar más rango dinámico en el paso 4, se transformará a una respuesta perceptible más próxima a lo que ocurre en la realidad. Se aprecia como las imágenes lineales anteriores parecían más oscuras 1.28.

Es importante que la función gamma concuerde con la pantalla en uso para obtener una respuesta predecible aunque, muchas veces, se emplean curvas diferentes con motivos artísticos para generar mas contraste o saturación. La OETF (Electro-Optical Transfer Functions) se refiere a una curva con una diferencia hacia las curvas gamma:

- Las funciones gamma son funciones exponenciales puras con una variable llamada gamma (lineal a Display = $pixel^{1/2,2}$, Display a lineal = $pixel^{2,2}$) y se muestra en la figura 1.27.
- Las OETF son curvas más elaboradas que no tienen que seguir de forma rigurosa un valor gamma. Muchas veces son referidas con una virgulilla como $\sim gamma2{,}2$. Este símbolo significa que son muy parecidas.

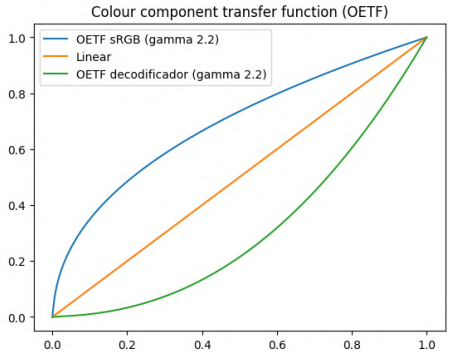


Figura 1.27: OETF sRGB

(a) sRGB 2.2 D65

(b) sRGB 2.2 D50

Figura 1.28: Imagen post procesada

Se ha mencionado que la OETF sRGB se emplea si no se ha realizado el paso de regeneración de blancos (paso 4). Esto se debe a que al realizar la fotografía, el usuario ha calibrado la escena con un punto de referencia llamado gris medio y normalizar un rango más grande que el previsualizado al tomar la imagen desplazaría hacia una zona más oscura este gris dejando al sujeto más oscuro de lo calibrado al tomar la fotografía.

1.5. Imágenes Generadas Por Ordenador

En el caso de las **imágenes generadas por ordenador** este sistema se ve un poco diferente. En primer lugar, **no se esta trabajando con luz real** (fotones) y no hay frecuencias fotoeléctricas que separar en el sensor.

Para entender este sistema, por el momento, se asumirá que **el sensor de una "cámara virtual"** (termino con el que me referiré a el elemento que hace de cámara en los sistemas digitales) **no tiene limitaciones** y capta toda la información que le llegue exactamente como es en la simulación. Esto no significa que la cámara virtual obtenga un resultado perfecto (sin ruido o distorsiones) sino que **imprimirá el resultado exacto de la simulación** y por tanto, **la calidad final de la imagen depende solamente del motor de renderizado** y no de la cámara con la que se toma la foto. Como mención especial, es posible crear una simulación que funcione por frecuencias y espectros (Spectral Ray Tracer System).

Este resultado no se puede llamar RAW por los siguientes motivos:

1. No requiere de un demosaico.
2. Los valores almacenados son directamente los que se pueden representar.
3. Durante la generación de los colores, el motor parte de un espacio de color con su punto blanco en el que también se encuentran las texturas y demás elementos de la escena.
4. En ocasiones se le aplican múltiples procesados con curvas antes de ser almacenados.

En esencia, **si se esta simulando rayos de luz en un espacio cromático RGB**, siempre **se devolverá un valor dentro de este espacio**. Hay mucha confusión en este aspecto y se cree que ciertos formatos como exr o bmp almacenan la información raw, pero esto no es cierto, almacenan la información en Scene Refered (gamma lineal). Algunos formatos permiten conservar un mayor rango dinámico que tiene utilidad en procesos posteriores, pero bajo ningún concepto se trata de imágenes RAW o imágenes sin espacio de color.

1.6. Espacios de color

Un espacio de color se define con:

- **Modelo de color**: RGB (rojo, verde, azul), CMYK (cian, magenta, amarillo, negro), HSV (matiz, saturación, valor), entre otros.
- **Gamut**: Primarios, rango de colores que un espacio de color puede representar (representado como un triángulo en un diagrama CIE 1.29).
- **Punto blanco**: Punto que representa el color blanco en un espacio de color (D65 y D50 entre otros).
- **Ecuación de transferencia**: Es la ecuación que se utiliza para transformar los valores de color entre diferentes espacios de color. Cada espacio de color tiene su propia ecuación de transferencia (anteriormente se ha mostrado la matriz de transferencia de una cámara concreta).

Hay otros elementos que se pueden añadir a un espacio, por el momento se mantendrá de forma simplificada.

Normalmente se emplean gráficos xy que representa la cromaticidad pero no la luminosidad de los colores 1.29. En la medida de lo posible es recomendable mostrará las tres dimensiones y los efectos que se producen con figuras tridimensionales 1.30 para observar el comportamiento de las altas luces y las sombras. Este efecto no suele estar documentado y supone uno de los comportamientos más importantes en imágenes generadas por ordenador, la saturación y de-saturación de las altas luces.

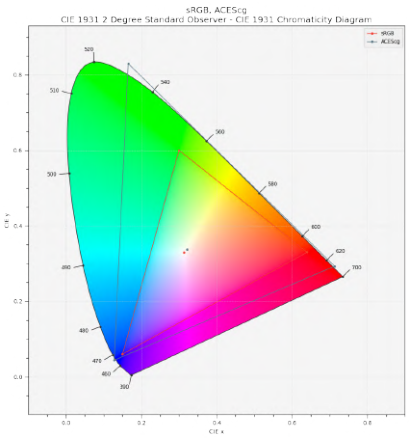


Figura 1.29: CIE 1931 chromatic diagram (sRGB D65, ACEScg D60)

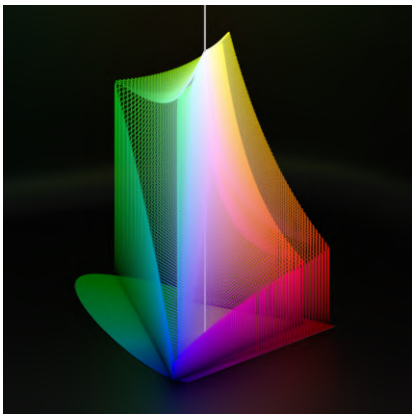


Figura 1.30: xyY chromatic diagram 3D (sRGB D65, ACEScg D65)

El sistema de tres primarios no es el único que existe y hay formas de representar todo el espectro visible con mucha más precisión y más de tres primarios, pero se trata de sistemas poco prácticos como laseres. El sistema RGB es empleado en todo tipo de pantallas que usan píxeles de colores “red, green, blue”.

1.6.1. Espacios frecuentemente usados

Algunos espacios de colores comunes son:

- **sRGB**: Empleado en muchas pantallas de ordenador, móviles y televisión cotidianas.
- **Adobe RGB (1998)**: Empleado durante la producción de algunas imágenes e ilustraciones.
- **ACEScg**: Empleado generalmente en software 3D para ser procesado por coloristas en otros software.

Normalmente no se puede elegir en que espacio de color se trabaja, este viene con el software. En el caso de blender, tiene por defecto sRGB D65. Otros sistemas mejor preparados para modificar espacios de colores como Davinci Resolve, trabajan en su propio espacio (DaVinci YRGB) que permite integrar algunos más pequeños. Una solución a este problema es usar sistemas de color management como OCIO.

Hay que tener en cuenta ciertos **factores al compartir imágenes en diferentes plataformas**. Cuando se declara **un espacio en múltiples aplicaciones puede que no utilicen las mismas características**. El caso de la curva sRGB es muy conocida. No se suele seguir un tipo de nomenclatura claro y estandarizado para diferenciar la OETF gamma 2.2 pura y la OETF sRGB. Revertir la transformación con la curva equivocada afecta las bajas y altas luces. Otro error menos frecuente es usar el punto blanco D50 en lugar del D65 estándar de sRGB dado que espacios como Adobe RGB lo emplean.

1.6.2. Como afectan los espacios de color en los motores de renderizado

El espacio de color con el que trabaja el motor de renderizado tiene un efecto directo en como se verán los colores. A diferencia de la conversión entre espacios, el motor genera valores que no corresponden al RGB original.

Para ilustrar este concepto, en la imagen 1.31 se muestra por columna las imágenes que deberían ser iguales pero, presentan resultados completamente diferentes, mientras que **la transformación entre espacios sí conserva la cromaticidad**. La transformación de sRGB, en la segunda fila, muestra como la transformación conserva los colores con alguna sutil diferencia en los verdes y amarillos del fondo por el punto blanco. Lo mismo ocurre con la primera fila.

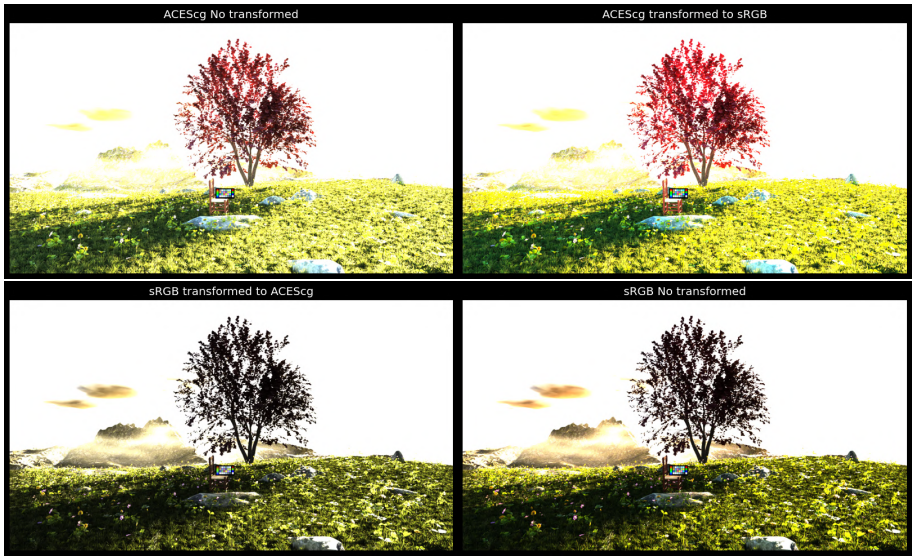
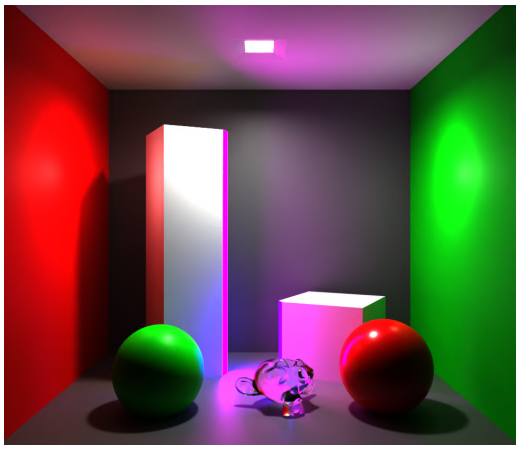


Figura 1.31: Comparación de espacios de color en gamma lineal

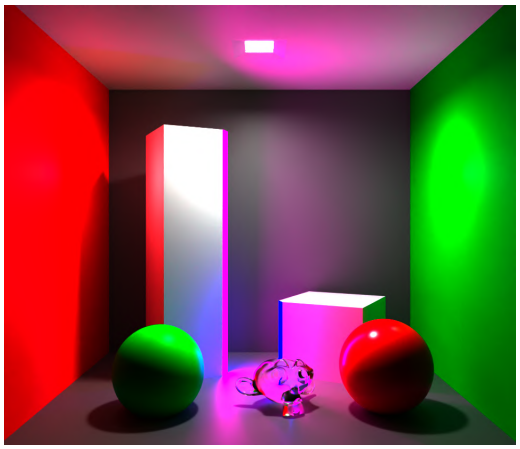
Esto ocurre porque **los colores empleados en el entorno 3D se declararon en el espacio sRGB y se transformaron a ACEScg antes de renderizar en ese espacio**. **Los máximos sRGB no se traducen como los máximos de ACEScg** 1.33. Los máximos verdes de sRGB se mostrarán como un verde medio en ACEScg, sin embargo, ambos verdes serán muy parecidos para el observador. Lo mismo ocurre con los otros colores como se mostró en el apartado de Traducción de colores reales a digitales.

Independientemente del efecto mencionado, si los colores están declarados de forma digital como un gradiente de color verde de luminosidad 0-1 (Valores no transformados entre espacios, sino

interpretados de forma absoluta por un espacio), el resultado seguirá presentando diferencias por como interpreta cada espacio los efectos cromáticos 1.32.



(a) Renderizado en espacio sRGB lineal y transformado a sRGB lineal.



(b) Renderizado en espacio ACEScg y transformado a sRGB lineal.

Figura 1.32: Diferencia de espacio de color empleada para renderizar valores cromáticos puros.

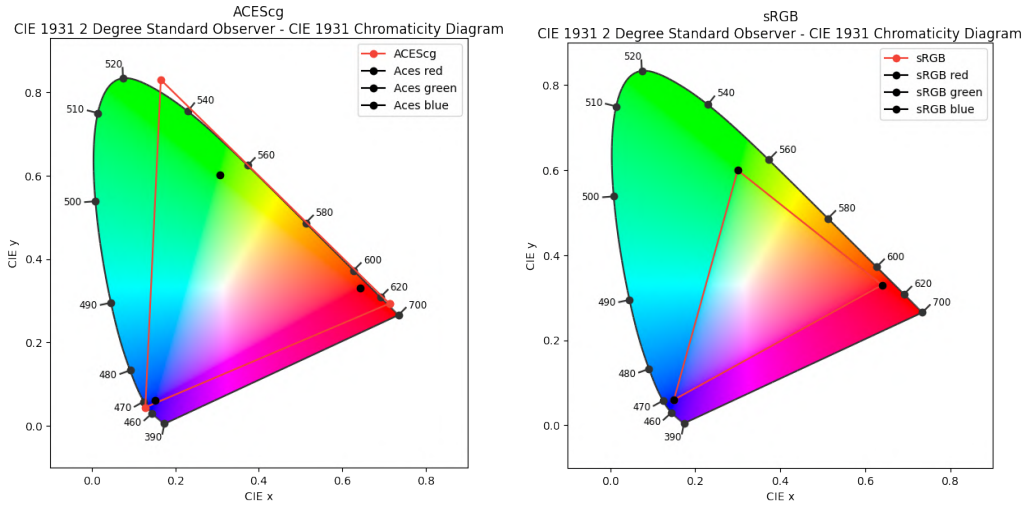


Figura 1.33: Máximos verdes de sRGB.

Esta es la razón por la que se defiende ACES como el nuevo espacio estándar para almacenar y editar imágenes. Los cálculos se realizarían sobre los colores reales que pueden ver nuestros ojos y luego serían transformables a espacios más pequeños como sRGB. Pero los espacios tienen otros efectos sobre el color, sobretodo cuando se trata de hacer cálculos nuevos sobre los colores.

También entra en juego el modo de interpretar la luminosidad de los canales. Como ya se ha explicado con anterioridad, esto se puede llevar a cabo con OETFs o LUTs. Que ocurre si una imagen ACEScg lineal se transforma a sRGB (incluida OETF) y una imagen sRGB lineal se transforma a sRGB(incluida OETF) 1.34?

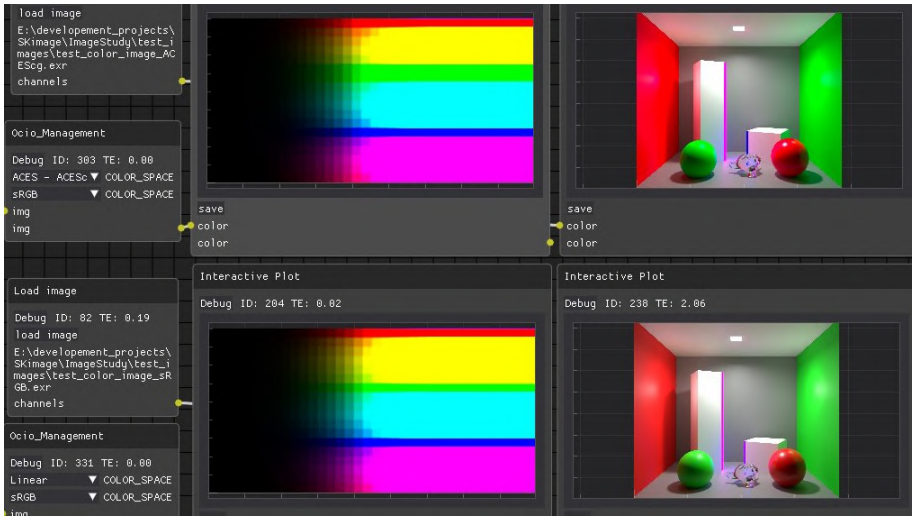


Figura 1.34: sRGB OETF en un gradiente de colores.

Los colores con intensidades de luz muy altas se congelan en un color (the notorius six [5]). Los primarios y los colores RGB se mantienen de forma pura. El efecto deseado suele ser que se transformen hacia blanco, y esto se puede conseguir con LUTs más complejos como el que usa blender 1.35.

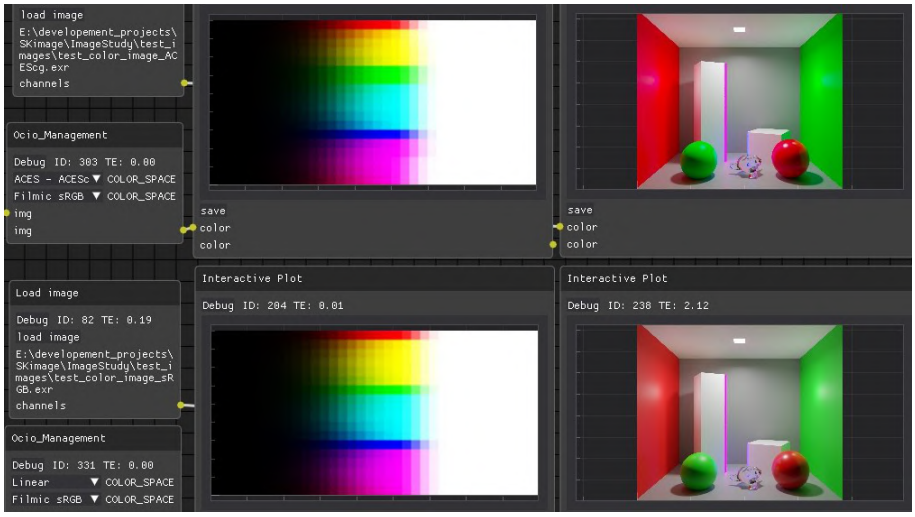


Figura 1.35: Filmic sRGB OETF en un gradiente de colores.

Se sigue observando que los colores terminan en los tres primarios CMY y los tres colores digitales RGB pero, tras cierto rango de luz, los colores se de-saturan. Este comportamiento no esta presente en todas las configuraciones de espacios de color como por ejemplo Nuke y spi-anim/vfx. También se observa como ACES tiende a saturar los colores más que el espacio sRGB.

El efecto que se puede llamar “Film” se consigue con un “camino hacia blanco”. Los pasos que siguen las configuraciones de color para conseguir un color blanco en altas luces suelen ser:

1. **Allocation**. Pasar el contenido lumínico Scene Referred a Display Referred (Tone mapping)

con una curva como log2 o Cineon.

2. **Lut o Matriz** para el camino hacia blanco. Este paso es diferente para cada configuración. Por ejemplo, Blender emplea un archivo filmic_desat65cube.spi3d, mientras que AgX emplea una matriz:

$$\begin{bmatrix} 0{,}842479062253094, & 0{,}0784335999999992, & 0{,}0792237451477643, & 0, \\ 0{,}0423282422610123, & 0{,}878468636469772, & 0{,}0791661274605434, & 0, \\ 0{,}0423756549057051, & 0{,}0784336, & 0{,}879142973793104, & 0, \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

3. Por ultimo se aplica una **OETF** como sRGB para añadir el contraste necesario para el display.

1.6.3. Gestión de color

La gestión de color es necesaria para trabajar con contenidos de múltiples procedencias. El objetivo es mantener una percepción de color igual o similar entre diferentes plataformas. Un entorno que emplee sRGB no puede interpretar datos en ACEScg sin una transformación previa. La correcta transición de un espacio a otro se consigue con el siguiente proceso (Suponiendo gamma lineal).

1. **De ACEScg a AP0**
   ACEScg esta compuesto por los primarios AP1. En la documentación oficial de ACES se muestra las matrices para pasar de AP1 a AP0 y AP0 a XYZ. Este es un paso extra con el espacio ACES, es posible pasar al siguiente paso si se conoce la matriz RGB_to_XYZ.

$$\begin{bmatrix} R_{AP0} \\ G_{AP0} \\ B_{AP0} \end{bmatrix} = \begin{bmatrix} 0{,}6954522414 & 0{,}1406786965 & 0{,}1638690622 \\ 0{,}0447945634 & 0{,}8596711185 & 0{,}0955343182 \\ -0{,}0055258826 & 0{,}0040252103 & 1{,}0015006723 \end{bmatrix} \cdot \begin{bmatrix} R_{AP1} \\ G_{AP1} \\ B_{AP1} \end{bmatrix}$$

2. **De AP0 a XYZ**
   De nuevo, esta matriz se encuentra en la documentación oficial.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}9525523959 & 0{,}0000000000 & 0{,}0000936786 \\ 0{,}3439664498 & 0{,}7281660966 & -0{,}0721325464 \\ 0{,}0000000000 & 0{,}0000000000 & 1{,}0088251844 \end{bmatrix} \cdot \begin{bmatrix} R_{AP0} \\ G_{AP0} \\ B_{AP0} \end{bmatrix}$$

3. **De XYZ a PCS**
   Para poder emplear muchos espacios de colores se suele establecer un PCS (Profile Connection Space). Un PCS muy empleado es XYZ y la transformación suele estar bien documentada. Se asume que las transformaciones son:

$$Espacio_x punto\_blanco_x -> XYZD65 -> sRGBD65$$

   Con este esquema, cualquier espacio de color de entrada pasará a XYZ D65 donde ya conocemos las transformaciones a sRGB por lo que tiene una implementación escalable.

Dentro del mismo espacio de color PCS (XYZ) podemos transformar el punto blanco ACES a D65.

$$[M] = [M_A]^{-1} \begin{bmatrix} \rho_D/\rho_S & 0 & 0 \\ 0 & \gamma_D/\gamma_S & 0 \\ 0 & 0 & \beta_D/\beta_S \end{bmatrix} [M_A] \qquad (1)$$

$$\begin{bmatrix} \rho_S \\ \gamma_S \\ \beta_S \end{bmatrix} = [M_A] \begin{bmatrix} X_{WS} \\ Y_{WS} \\ Z_{WS} \end{bmatrix} \qquad (2)$$

$$\begin{bmatrix} \rho_D \\ \gamma_D \\ \beta_D \end{bmatrix} = [M_A] \begin{bmatrix} X_{WD} \\ Y_{WD} \\ Z_{WD} \end{bmatrix} \qquad (3)$$

La matriz $[M_A]$ es una matriz de transformación de 3x3. $[M_A]$ presenta tres opciones dependiendo de la respuesta en los conos de los ojos. Los valores $\rho_D, \gamma_D, \beta_D$ y $\rho_S, \gamma_S, \beta_S$ se obtienen con los valores $X_D Y_D Z_D$ del punto blanco Destiny y $X_S Y_S Z_S$ del punto blanco Source.

Method	$[\mathbf{M_A}]$			$[\mathbf{M_A}]^{-1}$		
XYZ Scaling	1.0000000	0.0000000	0.0000000	1.0000000	0.0000000	0.0000000
	0.0000000	1.0000000	0.0000000	0.0000000	1.0000000	0.0000000
	0.0000000	0.0000000	1.0000000	0.0000000	0.0000000	1.0000000
Bradford	0.8951000	0.2664000	-0.1614000	0.9869929	-0.1470543	0.1599627
	-0.7502000	1.7135000	0.0367000	0.4323053	0.5183603	0.0492912
	0.0389000	-0.0685000	1.0296000	-0.0085287	0.0400428	0.9684867
Von Kries	0.4002400	0.7076000	-0.0808100	1.8599364	-1.1293816	0.2198974
	-0.2263000	1.1653200	0.0457000	0.3611914	0.6388125	-0.0000064
	0.0000000	0.0000000	0.9182200	0.0000000	0.0000000	1.0890636

Tabla 1.2: Tabla de métodos y sus matrices de conversión de espacios de color.

- ejemplo: $\rho_D, \gamma_D, \beta_D$ y $\rho_S, \gamma_S, \beta_S$

$$\begin{bmatrix} \rho_S \\ \gamma_S \\ \beta_S \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0{,}9533_{WS} \\ 1{,}0000_{WS} \\ 1{,}0088_{WS} \end{bmatrix}$$

Y la transformación completa de XYZ PCS es:

$$[M] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{0{,}95047}{0{,}9533} & 0 & 0 \\ 0 & \frac{1{,}0}{1{,}0} & 0 \\ 0 & 0 & \frac{1{,}08883}{1{,}0088} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = [M] \begin{bmatrix} X_{aces} \\ Y_{aces} \\ Z_{aces} \end{bmatrix}$$

4. Por ultimo, se pasa **de $\mathbf{XYZ}_{D65}$ a sRGB**

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = [M] \begin{bmatrix} 3{,}2404542 & -1{,}5371385 & -0{,}4985314 \\ -0{,}9692660 & 1{,}8760108 & 0{,}0415560 \\ 0{,}0556434 & -0{,}2040259 & 1{,}0572252 \end{bmatrix} \begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix}$$

   La matriz $[M]$ se debe recalcular para este paso con punto blanco D65 de XYZ y punto blanco D65 sRGB. No es necesaria porque el espacio PCS tiene el mismo punto blanco. Si se realizan los cálculos se obtiene una matriz identidad.

Entender el funcionamiento de una aplicación concreta, como Davinci, es necesario para una correcta gestión de color, esto implica transformar todos los datos que entren a este nuevo espacio como se ha mostrado arriba. Algunos programas permiten personalizar estos comportamientos y otros implementan APIs externas para una mayor flexibilidad. Los programas especializados en vídeo tienen un comportamiento muy preparado y muchas facilidades para el usuario por lo que los errores suelen crearse por fallos del usuario.

1.6.4. Rango Dinámico y Gamma

En las **escenas reales**, el **rango dinámico** es potencialmente **infinito** y para poder procesarlo primero se debe **comprimir en un rango de valores almacenable**.

En el caso de las imágenes raw, se almacenan en un numero determinado de bits (12-14 bits) que, junto al sensor, determinan el rango dinámico. El DR expresado en EVs es el mismo que la cantidad de bits de la codificación (en un caso perfecto donde el sensor puede captar toda la luz sin ruido) ofreciendo de 12 EV a 14 EV en algunas cámaras de alta gama 1.36. Un nuevo bit ocupa la mitad del rango dinámico. El nuevo bit se emplea para las altas luces..

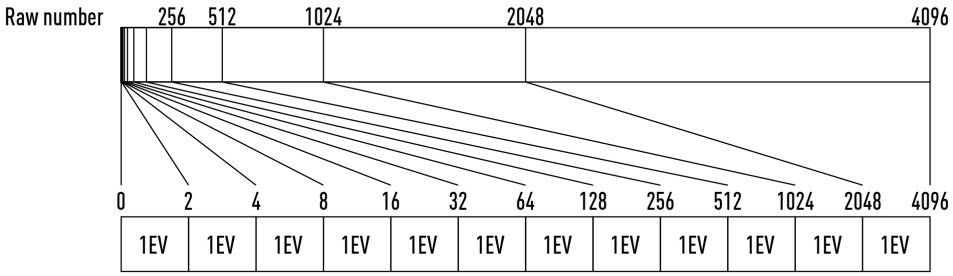


Figura 1.36: 12 bits repartidos en EVs [6].

En el caso de las imágenes generadas por ordenador, los valores representan información de luz real. En la figura 1.37 se muestra una imagen generada por ordenador. Tan solo los valores inferiores a uno son representables por pantalla y el resto se muestran como blanco puro. Se asume una profundiad de bits de float32.

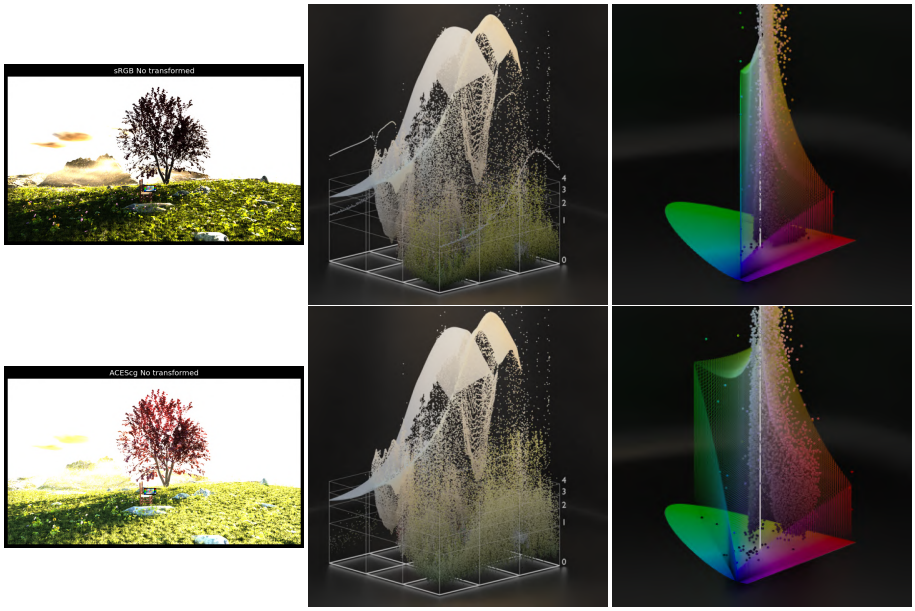


Figura 1.37: Luminancia representable vs Luminancia de la escena. A la izquierda: sRGB lineal, a la derecha: el valor lumínico de cada píxel.

1.6.4.1. Definición exacta

Rango Dinámico: diferencia entre el **valor máximo y mínimo de la señal**.

Esta descripción se puede emplear en cualquier ámbito, como el sonido o energía. En el caso de la fotografía, para que resulte más intuitivo, se habla de stops y EV.

1.6.5. Comportamientos importantes del rango dinámico

La discusión de que es el rango dinámico, pese a tener una definición muy clara, todavía genera malentendidos. En este documento ya se ha explicado como las imágenes raw tienen un **rango dinámico limitado por el sensor y por los bits del archivo final**. Este concepto suele llevar **erróneamente** a la conclusión de que el **rango dinámico depende de los bits de la imagen** y que una mayor cantidad de bits contiene un DR más grande.

En primer lugar, **las señales se pueden codificar con un numero de bits muy pequeño**. Con 2 bits se puede representar 4 valores diferentes, 0,1,2,3. **Si la señal original se compone de valores que oscilan entre 0 y 1000 se puede definir que**:

$$pixel = \begin{cases} 0, & \text{for } pixel = 0 \\ 1, & \text{for } 0 < pixel \geqq 500 \\ 2, & \text{for } 500 < pixel < 1000 \\ 3, & \text{for } pixel = 1000 \end{cases}$$

Es obvio que **el resultado será muy poco placentero a la vista pero** hemos almacenado un rango dinámico de 0 - 1000 de forma **correcta**. La razón por la que se **emplean una cantidad de 8**

Curva	Min	Max
sRGB	0	1
Filmic sRGB	$2^{-12,473931188}$	4.026068812
Filmic Log	$2^{-12,473931188}$	12.526068812

Tabla 1.3: Ejemplo de rango dinámicos

bits como mínimo, es **para preservar suficientes tono de grises** y evitar una visualización "fea" 1.38. En este caso, "fea" significa **evitar zonas de "bending"** donde el paso entre un gris a otro es demasiado visible. Este efecto es común en bajas luces mostradas por pantallas de televisión como cuando hay un fundido a negro y se pueden observar los píxeles de diferentes exposiciones. Otros factores también afectan a esta elección, como la forma binaria de operar en un procesador y una GPU, que están optimizados para bytes y múltiples de bytes como 8, 16 o 32.

Figura 1.38: sRGB codificado con seis tonos de grises

En el ejemplo de la **imagen raw**, la **imagen** esta **mostrada con** solo **8 bits**, pero el **rango dinámico** sigue siendo el **del sensor original**, la única diferencia es que preservamos menos tonos de grises.

Para referirse al rango dinámico, se emplearán valores Scene Referred absolutos que significan lo mismo en cualquier entorno. **Los EVs se emplearán cuando estemos editando la imagen** a partir de un punto de referencia como el gris medio.

Para evitar confusiones, en mi experiencia, es mejor no hablar de rango dinámico como pasos de luz excepto para fotografía. Para elementos digitales como CGI es más apropiado mencionar el valor mínimo codificado; el máximo; en que cantidad de bits se codifica, y si emplea una curva logarítmica para reposicionar los valores. Obsérvese como las siguientes imágenes poseen rangos dinámicos independientes a su profundidad de bits: 1.39 1.40 1.41.

Figura 1.39: $(0-1)$ **8bit con sRGB OETF**

Figura 1.40: $(2^{-12,4739} - 4{,}0260)$**, 8bit con Filmic sRGB OETF**

Figura 1.41: $(2^{-12,4739} - 12{,}5260)$**, 8bit con Filmic Log OETF**

La cantidad de bits es un factor importante para mantener suficientes tonos de grises en la manipulación de datos posterior. Para imágenes que solo pueden conservar valores de 0-1 como png, da igual la cantidad de bits del archivo ya que nunca los emplearán para conservar las altas luces, sino las sombras. Esto se observa en la imagen anterior 1.37 donde mas bits solo codificaran la informacion que se encuentra entre los valores 0 y 1. Por esta razón requerirían primero normalizar la información que se quiere conservar como por ejemplo dividir entre un valor como 27 para almacenar el DR de 0-27. A consecuencia de distribuir los bits ante una mayor tasa de grises, si el numero de bits es insuficiente se producirán artefactos en las zonas donde afecte mas la curva display empleada como se observa en las figuras 1.42 1.43 1.44.

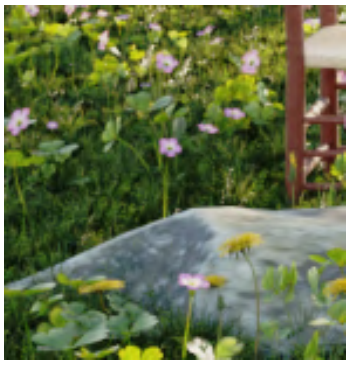


Figura 1.42: imagen de 32 bits lineal con una curva logarítmica aplicada

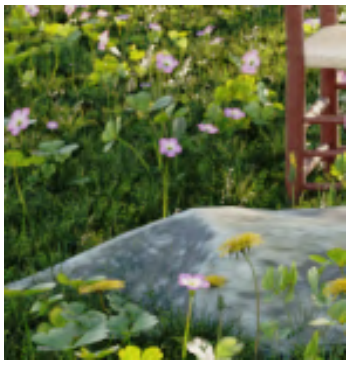


Figura 1.43: imagen de 16 bits lineal con una curva logarítmica aplicada

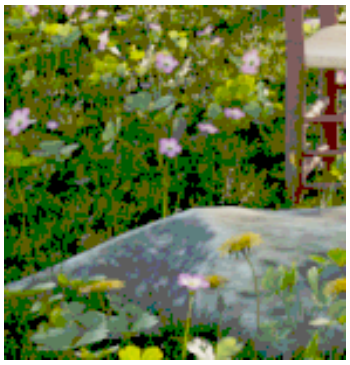


Figura 1.44: imagen de 8 bits lineal con una curva logarítmica aplicada

1.6.5.1. Usos del Rango dinámico

Para las imágenes **generadas por ordenador**, la compresión de la información varia **en función del uso**.

- **Imagen lista para Display referred**.
  Muchas veces, tras generar una imagen se busca **compartirla de forma directa** sin una edición posterior o con una edición llevada a cabo dentro del propio motor. Mientras la información permanezca en el software que genera la imagen, se suele trabajar con una profundidad de bits muy grande que mantiene suficientes valores para cualquier modificación como aumentar o disminuir la exposición.

  Al **exportar** la imagen, se **elige un espacio de color** con el que se va a almacenar y que DR se va a conservar. **La OETF** (Electro-Optical Transfer Functions) también **se aplica al codificar** en algunos formatos estándar como .png y .jpg. Ciertos formatos, como los mencionados anteriormente, no pueden almacenar valores no representables. Los 255 tonos de grises de las imágenes sRGB almacenadas en 8 bits son los valores contenidos entre el margen 0-1 de un motor de renderizado. El rango dinámico entre 0-1 es muy identificativo de una imagen y posee mucha información útil, sin embargo, muchas veces se busca un resultado que ofrezca más rango en las altas luces. En las siguientes imágenes se observa como el rango 0-1 contiene suficiente información como para apreciar la escena 1.45 1.46.

Figura 1.45: sRGB png 8bit -1EV

Figura 1.46: sRGB png 8bit -5EV

- **Imagen para futuras manipulaciones**.
  Esto incluye imágenes preparadas para **integración CGI y composición**. Estos archivos requieren un **margen de error para ser manipuladas posteriormente**. Esto se suele traducir como conservar un gran rango dinámico. Hay 3 opciones:

  1. Mayor profundidad de bits.
  2. OETF especializada.
  3. Compromiso entre profundidad de bits y OETF.

  En el primer caso, el inconveniente es el **tamaño de archivo** A.2. Comparada a las otras alternativas, esta es la que **mejor rango dinámico y calidad de imagen** ofrece al igual que una mayor versatilidad a la hora de implementarlas con otros contenidos multimedia.

  La segunda opción, OETF especializada, se puede emplear en varios escenarios. Por lo general se usa para **redistribuir los valores originales de forma que conserva más detalles** en menos bits. Para ello se sacrifica una gran cantidad de valores empleados en las altas luces para repartirlos en las sombras. Estas curvas no están pensadas para ser placenteras a la vista y se suelen llamar logarítmicas.

  En la figura mencionada anteriormente 1.36, se muestra como la mitad de los valores de la imagen se conservan en las altas luces, esto significa que la mayor parte de los bits almacenan esa información. Por lo general, los ojos son más susceptibles a los cambios de luz, sobretodo de las sombras a las medias y altas luces. La diferencia entre los valores de luz más altos de la escena pueden llegar a ser imperceptibles por lo que no es una buena idea emplear los recursos limitados de la memoria en almacenar esa información.

  Por ultimo, la opción de **combinar** estos dos **métodos**. Ciertos **formatos de imagen están creados con este fin**, uno de ellos, DPX, permite seleccionar la profundidad de bit y al mismo tiempo el usuario puede aplicar una OETF antes de codificar o usar la curva logarítmica CINEON que acompaña este formato.

1.6.5.2. El uso de las OETFs y los perfiles log

OETF (Opto-Electric Transfer Function), es una función de transferencia empleada para codificar la luz de forma que las sombras obtienen una mejor resolución y no se observe "bending". También se emplean para comprimir altas luces dentro del margen 0-1 conocido como "tone mapping". Las cámaras suelen emplear una **Camera encoding OETF como rec.709** que no

es igual a OETF rec.709. Tras este proceso se puede aplicar la EOTF para operar en un entorno Scene Referred (gamma lineal). Para mostrar el resultado por pantalla, de nuevo se aplica una OETF (esta vez la escogida para la plataforma display como una pantalla sRGB con OETF gamma 2.2). Este contenido es procesado por la pantalla aplicando la EOTF (Función de transferencia inversa de la OETF) que transforma el contenido a un espacio lineal emitiendo luz entre 0 y 100 nits.

Ciertas OETFs como sRGB OETF se emplean para dejar el contenido lumínico listo para la visualización mientras que otras **se emplean para conservar más información** en caso de necesitarla posteriormente **como las curvas log**. Este ultimo caso se muestra en las imágenes 1.49 donde reorganizan los valores de forma que se conserven mejor el rango dinámico. No todas son agradables visualmente y parece que carecen de contraste. A la hora de recuperar los valores originales de la escena (lineales), la diferencia de codificar con una curva logarítmica (log2 o CINEON) a una Display Refered (sRGB OETF) es notable como se muestra en 1.48 y la figura en el apartado superior 1.44.

Las OETF con el objetivo de Display Referred suelen estar pensadas para poca profundidad de bits y para dar un **resultado placentero** que muchas veces se expresa como un **gradiente perceptualmente lineal** 1.47. El gradiente Lineal no se percibe de forma correcta por nuestros ojos. El paso del negro puro a gris es demasiado grande y la diferencia entre los cinco últimos gradientes es demasiado pequeña. Los otros gradientes parecen seguir una sucesión más perceptiblemente lineal.

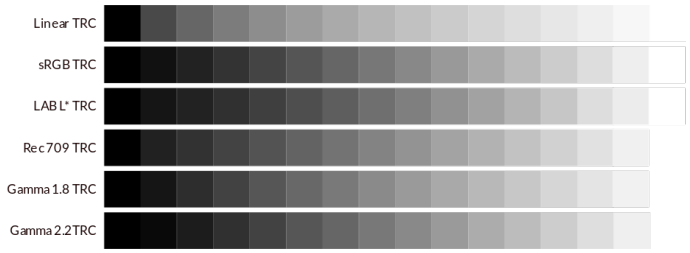


Figura 1.47: Gradiente lineal con diferentes OETFs con objetivo Display Refered.

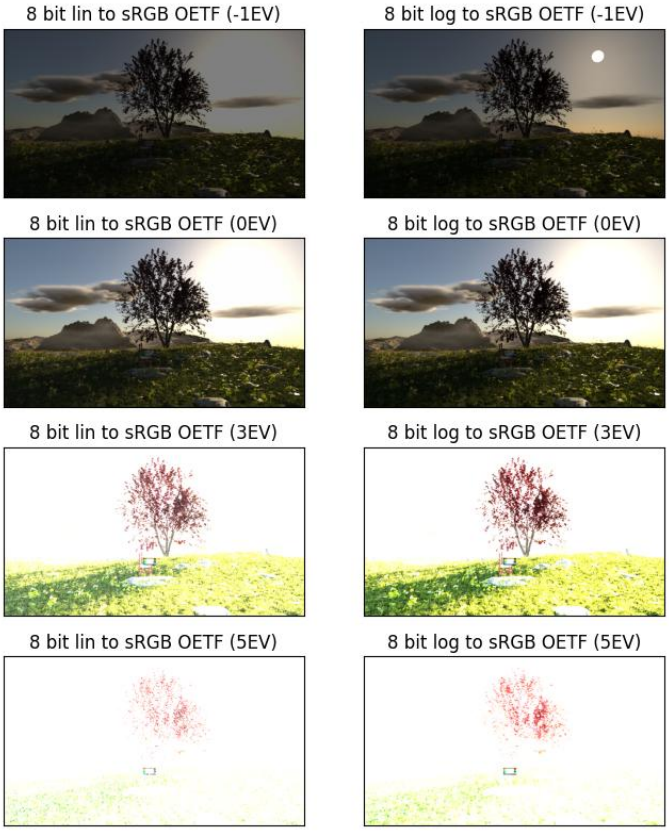


Figura 1.48: Diferencia al modificar la exposición de una imagen lineal y otra con codificación logarítmica

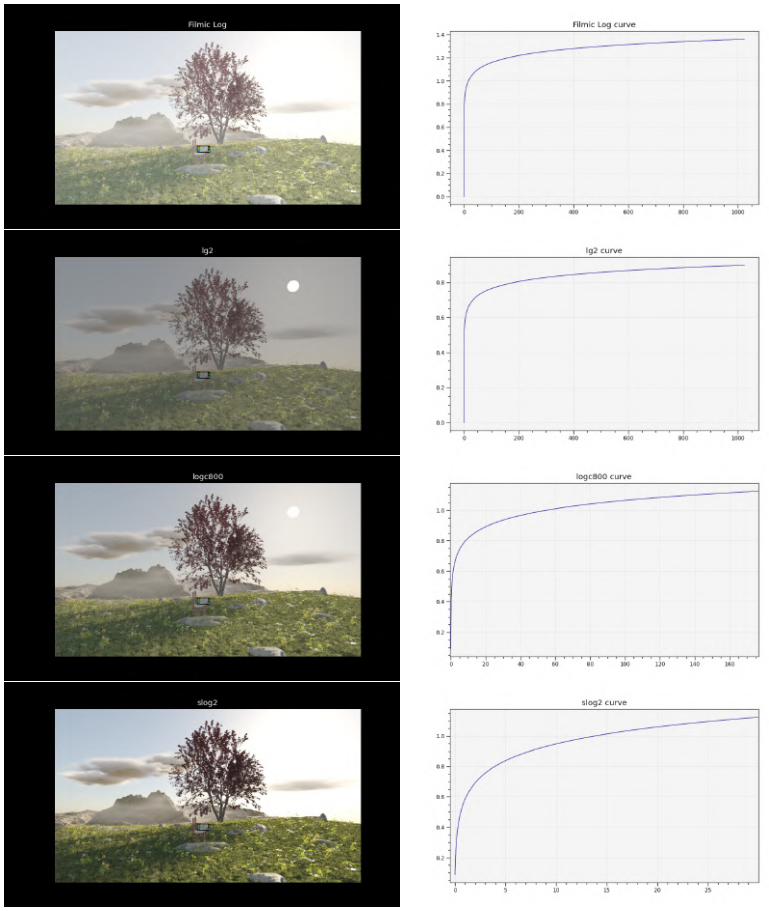


Figura 1.49: Diferentes OETFs log

Si has oído hablar de **OETFs y LUTs** es posible que te confunda la diferencia entre los términos. Las **OETFs son funciones matemáticas** que se aplican a los valores de las imágenes para transformarlos desde el espacio de color lineal al espacio de color no lineal. Esto **incluye LUTs, LMT, Gamma y otras funciones**, como operaciones cube3D, para modificar colores. Por ejemplo, la función OETF utilizada en el espacio de color sRGB transforma los valores de entrada lineales según la siguiente fórmula:

$$V_{out} = \begin{cases} 12{,}92V_{in}, & \text{si } V_{in} \leq 0{,}0031308 \\ 1{,}055V_{in}^{1/2{,}4} - 0{,}055, & \text{si } V_{in} > 0{,}0031308 \end{cases} \tag{1.2}$$

Donde C representa el valor del canal de color en el rango [0,1].

Los LUTs son tablas que mapean los valores de entrada a los valores de salida correspondientes. Es decir, los LUTs son una especie de funciones que se utilizan para realizar ajustes en el color, el contraste, la saturación, y otros elementos. Es decir, un LUT podría utilizarse para ajustar los tonos de piel de una imagen o para dar un aspecto retro o “vintage”.

```
Version 1
From 0.000000 1.000000
Length 4096
Components 1
{
0.0000000000000E+0
1.3546595638040E-5
2.7119409093229E-5
4.0718490533262E-5
5.4343890219558E-5
6.7995658507426E-5
8.1673845846238E-5
...
9.9992193266844E-1
9.9996105911473E-1
1.0000000000000E+0
}
```

Figura 1.50: LUT $filmic\_to\_0-70\_1-03.spi1d$. Empleado para transformar de Filmic Log a Filmic sRGB con contraste neutral.

Las curvas complejas como la sRGB, la rec709 y otras muchas, **se almacenan en este tipo de archivos** para una programación mas simple y estandarizada. En caso de no disponer de los LUTs, se pueden emplear funciones gamma (2^{gamma}). Otras funciones más complejas se pueden almacenar en tres dimensiones con un .spi3d o .cube que tienen en cuenta todos los posibles colores de una muestra y actuan en consecuencia.

1.6.6. Open Color IO

OCIO [7], definida por ellos mismos, es una **solución completa de gestión de color** diseñada para la producción de películas con un enfoque en efectos visuales y animación por ordenador. OCIO proporciona una experiencia de usuario sencilla y consistente en todas las aplicaciones compatibles, permitiendo opciones de configuración sofisticadas en la parte trasera que son adecuadas para producciones de alta calidad. OCIO es compatible con la Especificación de Codificación de Color de la Academia (ACES) y es agnóstico en cuanto al formato de LUT, admitiendo muchos formatos populares.

Por norma general, **OCIO se configura a través de un archivo config.ocio**. Nuke, Blender y otros programas tienen su propio config con su espacio de render, luts y otras caracteristicas. Es recomendable emplear uno de estos config ya generados para evitar errores de conversión.

A continuación se muestra el config.ocio empleado por blender:

- **Cabecera**

```
# OpenColorIO configuration file for Blender
#
# Based on aces, nuke-default and spi configurations from OpenColorIO-Config
```

```
#
# Filmic Dynamic Range LUT configuration crafted by Troy James Sobotka with
# special thanks and feedback from Guillermo, Claudio Rocha, Bassam Kurdali,
# Eugenio Pignataro, Henri Hebeisen, Jason Clarke, Haarm-Peter Duiker, Thomas
# Mansencal, and Timothy Lottes.
#
# See ocio-license.txt for details.

ocio_profile_version: 2

search_path: "luts:filmic"
strictparsing: true
luma: [0.2126, 0.7152, 0.0722]

description: RRT version ut33
```

- **Roles**:
  Un **rol** es una etiqueta para un espacio de color.

```
roles:
  reference: Linear
  # Internal scene linear space
  scene_linear: Linear
  rendering: Linear
  # Default color space for byte image
  default_byte: sRGB
  # Default color space for float images
  default_float: Linear
  # Default color space sequencer is working in
  default_sequencer: sRGB
  # Distribution of colors in color picker
  color_picking: sRGB
  # Non-color data
  data: Non-Color
  # For interop between configs, and to determine XYZ for rendering
  aces_interchange: Linear ACES
```

  En una declaración como $reference : Linear$, **reference** es una variable interna del sistema ocio. Todos los pasos que requieran el espacio **reference** llamarán directamente al espacio Linear que se definirá posteriormente.

- **Display**

```
displays:
  sRGB:
    - !<View> {name: Standard, colorspace: sRGB}
    - !<View> {name: Filmic, colorspace: Filmic sRGB}
    - !<View> {name: Filmic Log, colorspace: Filmic Log}
    - !<View> {name: Raw, colorspace: Raw}
    - !<View> {name: False Color, colorspace: False Color}
  XYZ:
    - !<View> {name: Standard, colorspace: XYZ}
    - !<View> {name: DCI, colorspace: dci_xyz}
    - !<View> {name: Raw, colorspace: Raw}
  None:
    - !<View> {name: Standard, colorspace: Raw}
```

```
active_displays: [sRGB, XYZ, None]
active_views: [Standard, Filmic, Filmic Log, Raw, False Color]
```

El diplay hace referencia a el espacio de color de tu monitor. Algunos monitores emplean sRGB mientras que los proyectores pueden emplear DCIP3. Dentro de cada display se encuentran las View.! $< View >$ name: Standard, colorspace: sRGB mostrará la información aplicando la OETF mientras que una View Raw mostrará la información de forma lineal.

- **Colorspace**:

```
colorspaces:
  - !<ColorSpace>
    name: Linear
    family: linear
    equalitygroup:
    bitdepth: 32f
    description: |
      Rec. 709 (Full Range), Blender native linear space
    isdata: false

  - !<ColorSpace>
    name: sRGB
    family:
    equalitygroup:
    bitdepth: 32f
    description: |
      sRGB display space
    isdata: false
    to_reference: !<FileTransform> {src: srgb.spi1d, interpolation: linear}
    from_reference: !<FileTransform> {src: srgb_inv.spi1d, interpolation: linear}
```

Esta declaración de dos espacios funciona del siguiente modo. En primer lugar definimos el PCS (Profile Connection Space) que se llama ”Lineal”. A continuación, se declara un nuevo espacio como sRGB. Las instrucciones $to\_reference$ y $from\_reference$, como su nombre indica, transforman de este espacio al PCS y viceversa. Los archivos empleados para esta transformación son srgb.spi1d y srgb_inv.spi1d que contienen las instrucciones LUT.

Se puede dar el caso de emplear un espacio de color para pasar a otro. Este es el caso de Filmic log y Filmic.

```
  - !<ColorSpace>
    name: Filmic Log
    family: log
    equalitygroup:
    bitdepth: 32f
    description: |
      Log based filmic shaper with 16.5 stops of latitude, and 25 stops of dynamic range
    isdata: false
    from_reference: !<GroupTransform>
        children:
            - !<AllocationTransform> {allocation: lg2, vars: [-12.473931188, 12.526068812]}
            - !<FileTransform> {src: filmic_desat65cube.spi3d, interpolation: best}
            - !<AllocationTransform> {allocation: uniform, vars: [0, 0.66]}
    to_reference: !<AllocationTransform> {allocation: lg2, vars: [-12.473931188, 4.026068812],
            direction: inverse}

  - !<ColorSpace>
```

```
name: Filmic sRGB
family:
equalitygroup:
bitdepth: 32f
description: |
  sRGB display space with Filmic view transform
isdata: false
from_reference: !<GroupTransform>
    children:
        - !<ColorSpaceTransform> {src: Linear, dst: Filmic Log}
        - !<FileTransform> {src: filmic_to_0-70_1-03.spi1d, interpolation: linear}
```

Para emplear el espacio Filmic sRGB primero se transforma de Linear a Filmic log, y luego a Filmic.

1.6.7. Conclusiones

Las imágenes capturadas con sensores y las generadas a partir de simulaciones se comportan de forma diferente. El proceso para unificarlas no es único y depende de las cámaras y motores de renderizado.

Dado que la tecnología de los sensores y procesadores es limitada, la solución debe presentarse por los medios digitales y ofrecer alternativas a los motores de renderizado al igual que a las configuraciones de espacios de colores para adaptarse a las imágenes procedentes de otros dispositivos.

El proceso de renderizado separado en dos partes deberá realizarse del siguiente modo: en el primer paso, los colores se calcularán en un espacio de color que contenga todo el espectro visible. En el segundo paso se aplicará un DRT que transforme el contenido a una colección de colores que conserve la crominancia y de-saturacion de un modo controlable para generar un punto de partida igual para la manipulación posterior de colores.

Ahora solo queda procesar el material simulado con las características de un sensor. Las ventajas de este proceso es:

- Las imágenes CGI ocuparían el mismo espacio que las capturadas por cámara.
- El rango dinámico almacenado en el archivo CGI correspondería al material capturado por cámara.
- Durante la gradación de color, el comportamiento de ambos archivos sería igual.
- El procedimiento seria estándar para cualquier sftware ya existente

1.7. Formatos de imágenes

En la industria audiovisual, por lo general, se ha decidido emplear formatos de imágenes con una profundidad de bit alta (16 - 32 bits). Estos archivos tienden a pesar mucho incluso comprimidos. Entre ellos se encuentran los .png de 16 bit, .exr 16-32 bits, .tiff 16 bit y otros. La diferencia entre estos reside en lo que representa los valores almacenados en cada estándar.

Formato Imagen	color bitdepth	datos	color space
BMP	8	uint	OETF
Iris	8, 16	uint	
PNG	8, 16	uint	OETF
JPEG	8	uint	OETF
JPEG 2000	8, 12, 16	uint	OETF
Targa	8, 15, 16, 24, 32	uint	OETF
Cienon	8, 10, 12, 16	uint	linear
DPX	8, 10, 12, 16	uint	linear
OpenEXR	16, 32	float y float half	linear
Radiance HDR	8	uint	linear
TIFF	8, 16	uint	OETF
WebP	8	uint	OETF

Tabla 1.4: Formatos de imágenes

1.7.1. BMP

BitMap [8].

1. Variedad de canales: Grey, RGB.
2. Compresiones admitidas: Sin compresión.
3. Ámbito de uso: Formato creado por microsoft para pasar imágenes entre plataformas.

1.7.2. Iris

Obsoleto, no se profundizará.

1.7.3. PNG

Portable Network Graphics [9].

1. Variedad de canales: Gray, RGB, RGBA.
2. Compresiones admitidas: Deflate algorithm (sin perdidas).
3. Ámbito de uso: Empleado en internet y para imágenes que requieren canal de transparencia.
4. Esta extensión tiene alteraciones como APNG para almacenar animaciones.

1.7.4. JPEG

Joint Photographic Experts Group [10].

1. Variedad de canales: Grey, RGB.
2. Compresiones admitidas: JPEG (con perdidas).
3. Ámbito de uso: almacenar imágenes con poco peso y que no requieren transparencia.

1.7.5. JPEG 2000

Joint Photographic Experts Group 2000 [10].

1. Variedad de canales: RGB, RGBA.
2. Compresiones admitidas: JPEG 2000 (JP2) (sin perdidas), JPEG 2000 (JP2) (con perdidas).
3. Ámbito de uso: No es usado regularmente y es soportado por pocas aplicaciones.

1.7.6. Targa

Truevision Advanced Raster Graphics Adapter [11]. Se puede abrir con Adobe ilustrator.

1. Variedad de canales: RGBA.
2. Compresiones admitidas: sin compresión.
3. Ámbito de uso: Adobe ilustrator y otras aplicaciones adobe.

1.7.7. Cienon

Cineon [12].

1. Variedad de canales: RGB.
2. Compresiones admitidas: ...
3. Ámbito de uso: Empleado para almacenar “scans”de film real. Se codifica con una curva log. En mi experiencia, no resulta útil para almacenar altas luces pero si sombras.

1.7.8. DPX

Digital Picture Exchange [13].

1. Variedad de canales: RGB, RGBA.
2. Compresiones admitidas: Codificación de longitudes de onda (RLE), sin compresión.
3. Ámbito de uso: Empleado para almacenar “scans”de film real. Posibilidad de codificación logarítmica.

1.7.9. OpenEXR

Nombrado como OpenEXR o EXR [14].

1. Variedad de canales: RGB, RGBA, indeterminado.
2. Compresiones admitidas: ZIP, ZIPS, PIZ, RLE, PXR24, B44, B44A, DWAA, DWAB.
3. Ámbito de uso: almacenamiento y distribución de CGI.

El formato EXR es el más empleado en entornos de imágenes generadas por ordenador debido a una gran precisión de punto flotante y a la versatilidad en sus canales de colores. A diferencia de los formatos de imagen mencionados anteriormente, EXR permite un numero N de canales y posee su propio método para etiquetar y agruparlos. Un ejemplo de este comportamiento seria el almacenar un archivo con 4 canales para el color (RGBA) en el padre **color** y otro padre con componentes RGB para almacenar la dirección de los normales de una escena 3D. La imagen estaría compuesta de 7 canales repartidos en dos grupos y se convertiría en una multilayer EXR. No todos los programas leen los multilayer EXR pero si suelen implementar EXR con canales RGB y RGBA.

Para mostrar los diferentes tamaños de archivo se ha renderizado una animación de 150 fotogramas exportada a múltiples formatos A.1 y A.2

1.7.9.1. Cualidades

El formato .exr es almacenado con valores de tipo float32 y floatHALF (16 bits). Los valores binarios son posteriormente interpretados con 1.51.

$$(-1)^{signbit} * 2^{exponent-15} * (1 + \frac{significand}{1024}) \tag{1.3}$$

Figura 1.51: floatHALF

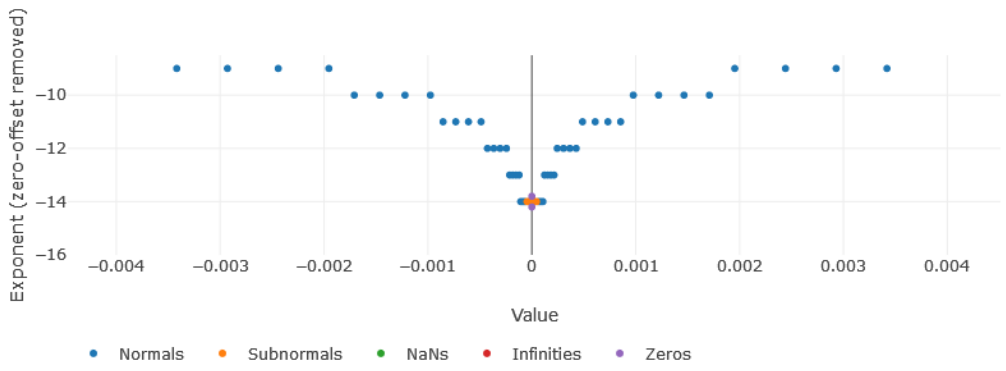


Figura 1.52: float half resolution [15]

Este sistema tiene un efecto en los valores almacenados 1.52. Los valores presentan una mayor resolución al estar mas cerca de cero. Se reparten más tonos de grises en las sombras que en las altas luces. Este comportamiento es el obtenido al emplear curvas logarítmicas. Por esta razón es importante **nunca** emplear OETFs antes de codificar un EXR, de lo contrario perderíamos las ventajas de este formato.

1.7.10. Radiance HDR

Formato normalmente con extensión .hdr [16]. Compuesto de píxeles de 8 bit por color con 4 canales (RGBE). El canal E, con 8 bits por píxel, presenta el exponente de la luz.

1. Variedad de canales: RGBE, XYZE.
2. Compresiones admitidas: RLE.
3. Ámbito de uso: Imágenes de alto rango dinámico para pantallas HDR y entornos digitales empleado para emisión de luces.

1.7.11. TIFF

Tagged Image File Format [17]. Compresión sin perdidas.

1. Variedad de canales: Gray, RGB, RGBA.
2. Compresiones admitidas: CCITT T.4 bi-level encoding, CCITT T.6 bi-level encoding, LZW, JPEG.
3. Ámbito de uso: Suele ser empleado para imágenes que requieren alta fidelidad y profundidad de bits como texturas.

1.7.12. WebP

WebP [18].

1. Variedad de canales: RGB, RGBA.
2. Compresiones admitidas: Arithmetic entropy encoding.
3. Ámbito de uso: Creado por Google para imágenes web. Compresión alta sin perdidas.

1.7.13. Elección de un formato

Los formatos expuestos no son los únicos que existen pero, abarcan una gran variedad de usos que ayuda a ejemplificar el comportamiento de las extensiones.

El objetivo de elegir un **formato adecuado es preservar la información útil en el menor espacio posible** y el hecho de que existan tantos formatos es porque hay una gran variedad de necesidades.

Primero se debe determinar el uso que se le va a dar a la imagen, es decir, si es para mostrar o editar. Los formatos más pesados almacenan información que no suele ser representable.

Otro factor determinante son los **canales almacenables**, sobretodo en el caso de requerir más de 4 canales de color o si se requiere de un canal de transparencia.

Es muy fácil ser atraído por la idea de **emplear EXR por su gran cantidad de grises**, sin perdidas y canales indefinidos, pero esto puede **llevar a proyectos con tamaños superiores a 200 GB** con solo 5 minutos de metraje y que además resultan muy difíciles de procesar con softwares como Davinci Resolve. Hay alternativas que pueden manejar el rango dinámico necesario sin 16 bits. Si las imágenes EXR deben componerse con metraje obtenido por cámaras reales, todo el rango dinámico almacenado quedará inservible al no poder combinarlo con el material original.

La elección de un formato requiere conocer el uso de las imágenes finales, el espacio en memoria del que se dispone y las propiedades necesarias que se quieren conservar.

1.8. Tecnología display

Como se puede intuir, las tecnologías existentes para mostrar imágenes son muchas y muy variadas. Se pueden encontrar monitores y proyectores con múltiples tecnologías. Todas estas variables se combinan entre ellas y forman un sistema de display que requeriría de un estudio independiente para hacer contenido especifico a este método de reproducción.

Puesto que el objetivo de este trabajo es ofrecer una vista general para poder investigar de forma independiente los casos personales, se explorará los casos más comunes y la tecnología que comparten diferentes displays.

1.8.1. Monitores y pantallas

En este grupo se considerarán pantallas móviles, monitores, televisores y este tipo de tecnología.

Estos dispositivos de salida se forman con píxeles. La tecnología de cada píxel puede variar en OLED, AMOLED, Super AMOLED, Dynamic AMOLED, LCD, LED, CRT, Plasma, DLP, entre otros.

Al mismo tiempo, la luz emitida por una pantalla se mide en **NITS**. Un nit corresponde a una candela por metro cuadrado. Las pantallas convencionales poseen unos 100 nits. Dependiendo de la tecnología del píxel, el valor más negro que puede representar una pantalla puede ser unos 0.001 nits. Si una imagen conservara valores más pequeños que este nivel de negro, no podrían ser representados. El valor negro puro solo se puede mostrar en pantallas como la AMOLED que permiten el cero absoluto.

Los primarios de la pantalla vienen definidos por la tecnología que empleen. Para saber que colores puede representar cada dispositivo se deben consultar sus especificaciones.

1.8.1.1. Tecnología HDR

Las pantallas HDR solo funcionan con contenido HDR. En el mercado, las tecnologías dominantes son Dolby Vision y HDR10. Se trata de contenido entre 10-12 bits con primarios BT.2020.

- Dolby Vision ha desarrollado su propio formato HDR de forma privada con una curva PQ OETF (Perceptual Quantizer). Funciona con una profundidad de 12 bits y metadatos que indican como mostrar cada escena.
- HDR10 es un estándar abierto de 10 bits. Emplea metadatos estáticos que modifican las luces una única vez durante toda la transmisión.
- HDR10+. Estándar abierto de 10 bits con metadatos dinámicos.
- Technicolor. Formato creado en europa para ofrecer retro-compatibilidad de contenido HDR para mostrarlo en SDR y viceversa.
- HLG (Hybrid Log-Gamma). Planteado como perfil retrocompatible enfocado a televisores.

Si el rango dinámico de una imagen no depende de la profundidad de bits y el objetivo de una pantalla HDR es mostrar un rango dinámico más amplio, para que se emplean 10 bits?.

La OETF PQ (EOTF: SMPTE ST 2084 (PQ)) empleada por HDR10, HDR10+ y Dolby Vision, representa muy bien el comportamiento de nuestros ojos ante la luz emitida. HDR10 sostiene que diez bits es suficiente para no percibir "bending"(tonos de grises muy separados) mientras que Dolby emplea 12bits 1.53. Esta curva empleada por los dispositivos HDR no es compatible con la curva rec709 empleada por pantallas SDR. Realizar dos gradings independientes es común para adaptar la imagen a los diferentes requisitos de HDR y SDR.

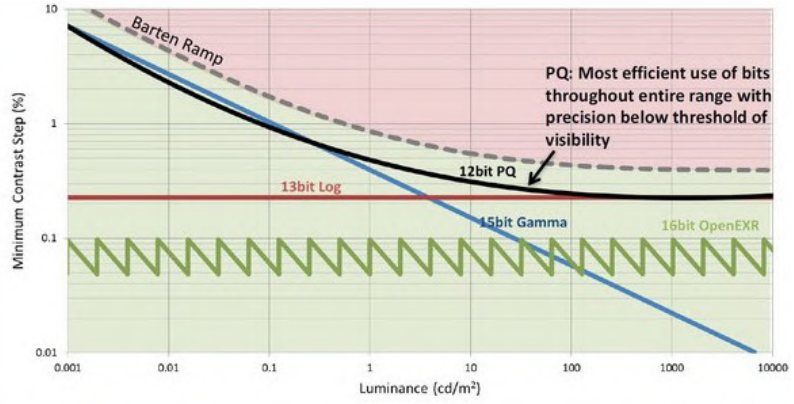


Figura 1.53: Dolby curva de bending en función de los bits [19]

El proceso es el mismo empleado que el usado en procesado raw. La imagen capturada debe componerse de valores entre 0 y 1024. Los datos Scene Referred deben transformarse por medios de LUTs o otras funciones para comprimir la luminancia.

1.9. Un nuevo Color Management Open Source

Tras la investigación anterior, queda claro que el sistema de color empleado por las aplicaciones no es el mejor ni el más ajustado a los usuarios, es mas bien una solución que funciona. Para este trabajo se explorará una solución más completa y customizada para usuarios de efectos visuales y CGI.

Para el espacio de renderizado digital se empleara ITU Rec.2020 por las siguientes razones:

- Produce los resultados más próximos a un motor de renderizado espectral. Se pueden consultar los resultados en los experimentos realizados por Anders Langlands [20]. Donde más falla es en las tonalidades rojas al igual que ACEScg.
- El rango de colores es perceptiblemente igual a ACEScg y no contiene valores fuera del diagrama CIE 1931.
- Los primarios Rec2020 son los empleados en las pantallas HDR y son más amplios que en el espacio P3 empleado en cine.

Para el Display Render que transforma la información Scene Referred ITU rec.2020 en la seleccionada por el usuario se empleará una solución basada en la propuesta de Jed Smith, OpenDRT [21].

- Transformación de gamut de entrada (rec.2020) a gamut de salida (rec.709, ACES, ...)
- Tone Scale basada en una propuesta de Daniele Siragusano en los foros ACES (Daniele Curve)
- Gamut Compresión hacia blanco. Este paso solo se lleva a cabo en la configuración ocio de blender y en la configuración ocio de TCAM (FilmLight). Es un proceso crucial que suele tener varias opciones. Para este experimento se realizará una transformación que preserve una crominancia lineal 1.54 1.55, 1.56.

Estos dos procesos generan una imagen como punto de partida para un LUT que simulará el proceso de una cámara especifica para transformar los colores.

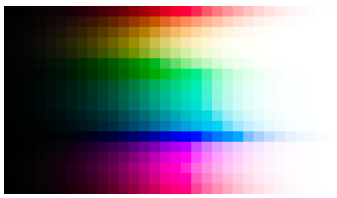


Figura 1.54: ACEScg primaries - OpenDRT

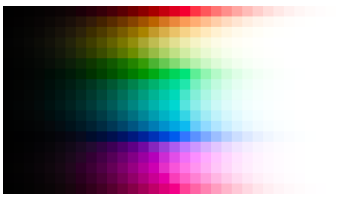


Figura 1.55: rec.2020 primaries - OpenDRT

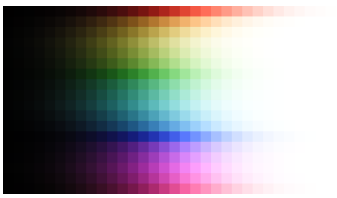


Figura 1.56: rec.709 primaries - OpenDRT

Para poder implementar este conjunto de funciones de forma estandarizada se creará un LUT con unas funciones generadas en python que recreen este efecto. Para generar este LUT se siguen los pasos:

- Generar una tabla de valores lineales que representan los colores en forma de cubo. Este tipo de archivo se llama .spi3d y contiene todas las posibles combinaciones de los clores con 65 muestras por color.
- De Display Referred a Scene Referred. Se trata de el paso opuesto a la hora de renderizar.

$$SceneReferred_{val} = 2^{(Tabla_{rgb} * (max_{val} - min_{val}) + min_{val})} \tag{1.4}$$

Donde $max_{val} = 12{,}539267881710824$ dado que el valor máximo representable por OpenDRT antes de el blanco puro es $2^{12{,}539267881710824}$ y $min_{val} = 2^{-12{,}473931188}$.

- Transformación DRT a partir de los valores lineales $SceneReferred_{val}$. Este valor se almacena dentro de la tabla original $Tabla_{rgb} = f(SceneReferred_{val})$.

Tras generar el LUT, se realiza una configuración ocio que realiza las siguientes operaciones:

- XYZ a REC2020.
- Transformación logarítmica normalizada entre valores min y max.

$$normalized_{val} = (log_2(SceneReferred) - min_{val})/(max_{val} - min_{val}) \tag{1.5}$$

- Transformación LUT creada para este proyecto.
- El resultado requiere de una curva como la sRGB para mostrarse de forma placentera.

Este nuevo sistema esta disponible en la aplicación desarrollada en python para que cualquiera lo pueda probar. Los mejores reflejos y materiales emisivos se obtienen de OpenDRT 1.59 mientras que “la sensación de luminosidad” se aprecia mejor en los otros DRT. 1.57 1.58 1.60

Figura 1.57: rec.2020 primaries - filmic sRGB

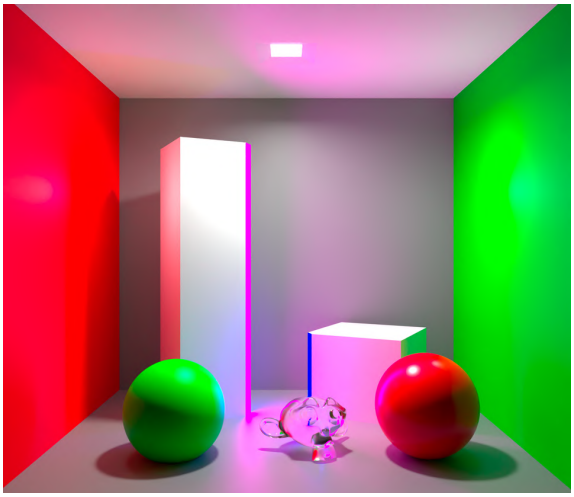


Figura 1.58: rec.2020 primaries - sRGB

Figura 1.59: rec.2020 primaries - OpenDRT sRGB

Figura 1.60: rec.2020 primaries - Aces sRGB

Capítulo 2

Software para estudio de imágenes y color

La segunda parte de este proyecto se basa en la creación de una aplicación que rellene las carencias de algunos softwares. Este programa ha sido empleada en algunos apartados anteriores.

Las aplicaciones suelen estar especializadas en un campo y **requieren de otros programas para completarse**. Un claro ejemplo es el de algunas aplicaciones como blender que no permiten cargar imágenes raw o aplicaciones como Davinci Resolve hacen un renderizado raw automático. Por lo general se emplearía lightroom o photoshop para exportar la imagen en un formato adecuado.

Otro problema es la **muestra y comparación de múltiples imágenes de forma cómoda e intuitiva**. Para este tipo de acciones se suele emplear Nuke que permite cambiar rápidamente entre imágenes y funciona únicamente con un sistema de nodos. Incluso con este funcionamiento no es lo más útil para el objetivo presentado en este trabajo y la **curva de apredizage es muy elevada dado que es un software planteado para profesionales y personal especializado**.

Por último, algunas aplicaciones existentes como blender y nuke **permiten la implementación de códigos en lenguaje python pero con sus propias APIs y mezclándolo con sus propios sistemas internos**. En la aplicación desarrollada completamente en python la integración de nuevos módulos es tan fácil como escribir en python puro. Esto presenta una ventaja para aquellos usuarios que busquen mucha versatilidad y un desarrollo rápido y simple.

2.1. Requisitos de una aplicación

La aplicación que se diseñará a continuación debe cumplir las siguientes condiciones:

- Study Image debe pesar poco y ser rápida de ejecutar. El objetivo es evitar emplear otros programas más pesados con muchas funcionalidades que no se requieren para estudiar propiedades de las imágenes y los colores (Nuke, Davinci Resolve, photoshop...).
- Visual e intuitiva. Para evitar confusiones, debe mostrar paso a paso los procesos que lleva a cabo el usuario para ayudar a entender el procedimiento.
- Implementar OCIO. No es necesario emplear esta librería para hacer una correcta gestión

de color pero, dado que es open source y utilizada por todos los programas mencionados anteriormente, ofrece la oportunidad de gestionar color a través de diferentes plataformas. El objetivo es poder crear y manipular una configuración ocio que puedas exportar a otras aplicaciones.

- Código simple y manipulable. Esta aplicación ofrece el código para poder crear módulos en función de las necesidades del usuario. Se sacrificará un poco de rendimiento con intención de facilitar la comprensión del código.

2.2. Estructura interna

Image Study es una aplicación de nodos. El usuario puede crear códigos más complejos a partir de pequeñas cajas con funcionalidades muy concretas. Se ha elegido este método por ser un sistema empleado en una gran variedad de aplicaciones (blender, nuke, davinci resolve, fusion, unreal engine, unity).

La aplicación esta desarrollada completamente en python para una mayor accesibilidad a personas que no sepan mucho de programación y que quieran entender el código interno.

Los nodos son hijos de una clase que controla las conexiones, actualizaciones y datos. Con este método, añadir un nuevo nodo es tan fácil como declarar un hijo, su función y su nombre.

```
study_image
├── modules
│   ├── __init__.py
│   ├── color_transform.py
│   ├── debugger.py
│   ├── father_class.py
│   ├── interaction.py
│   ├── loadders.py
│   ├── math.py
│   ├── operators.py
│   └── plotters.py
├── colormanagement
└── app.py
```

2.3. Ejecución de nodos

Cuando se **desarrolla una aplicación de nodos** se tienen ciertas características en cuenta. Por norma general, se suele emplear un lenguaje que soporte **asincronia**. Se crea uno o **varios nodos de entrada y un único nodo de salida** y se **ejecutan de fin a inicio** llamando cada vez a los nodos anteriores requeridos. Las **ventajas** de este método es que solo **ejecuta los nodos necesarios para la salida** y no lineas de nodos que no terminen en el nodo salida. La **desventaja** de este método, si se hubiera implementado en Study Image, seria que Study contiene **múltiples nodos output** y no uno único desde el que controlar todo el árbol 2.1. Esto supone una dificultad añadida al diseño puesto que se debería diferenciar entre tipo de nodos y no tratarlos a todos como iguales. Otra desventaja sería que **requiere ejecutar todos los nodos cada vez** que se manipule una nodo

intermedio.

Por comodidad, se ha **elegido un método síncrono** que ejecuta **de inicio a fin** 2.2. Esto permite que cualquier ramificación del árbol se ejecute. Para **evitar calcular de nuevo nodos que no han sufrido un cambio**, cada nodo almacena en sus salidas el valor ejecutado, y la ID del nodo se almacena en una lista de nodos ya ejecutados. Si un nodo se modifica, el, y todos los nodos conectados por delante, desaparecen de la lista y se vuelven a ejecutar 2.3.

Estos métodos están diseñados de modo que **no se ejecuta ningún nodo dos veces en una misma cadena**. También se evita ejecutar un nodo que tiene entradas de nodos no ejecutados todavía 2.4.

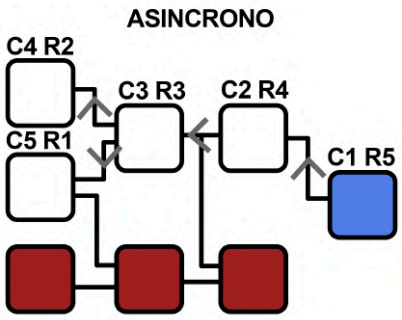


Figura 2.1: Llamada de fin a inicio. Donde C = call y R = result.

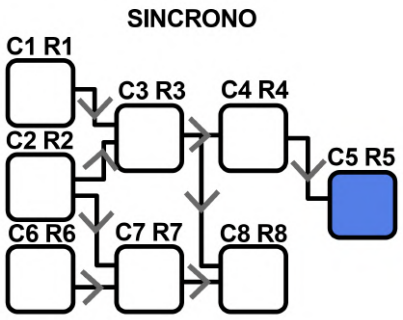


Figura 2.2: Actualización de nodos hacia delante. Todos los nodos tienen el mismo valor jerárquico.

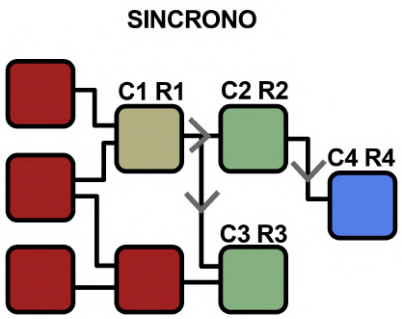


Figura 2.3: Actualización hacia delante a partir de un nodo modificado.

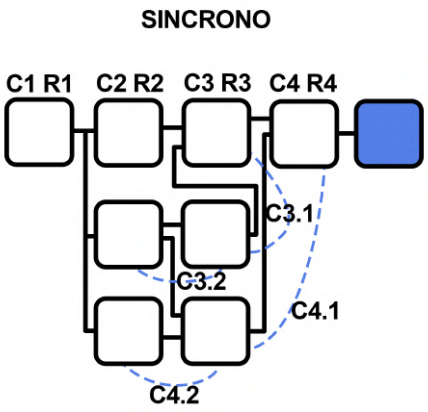


Figura 2.4: Solución a llamada síncrona que requiere nodos no ejecutados.

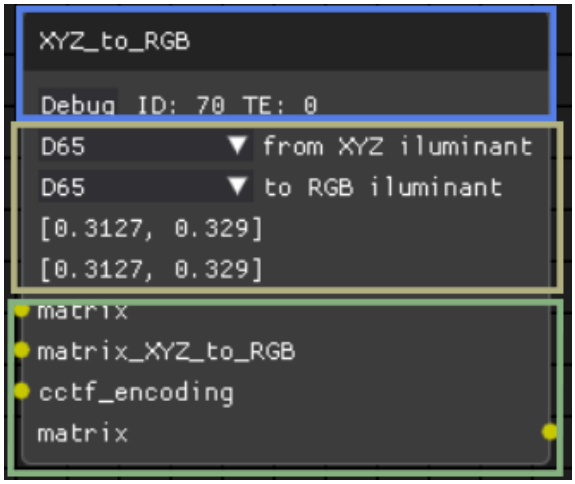


Figura 2.5: Las tres partes de un nodo: cabecera, static y pins

2.4. Creación de nuevos nodos

Los nodos siguen una estructura de tres partes 2.5; **Cabecera** (azul), **static** (amarillo) y **pins** (verde). Por otra parte, pueden contener información adicional que solo se puede acceder en el menú Inspector y con el nodo seleccionado. Con este diseño se pretende limpiar la pantalla de nodos y que no resulte sobrecargada para el usuario.

- Nombre: Al declarar un nodo se debe especificar el nombre con el que se representará. También controla un elemento común entre todos los nodos, la linea debug. Esta se compone del tiempo de ejecución, un botón con funcionalidades y el valor ID del nodo.
- Static: el nombre static viene por la variable que controla los elementos interactivos y estéticos de un nodo **self.static**. El usuario puede añadir todos los textos, botones, imágenes y elementos que crea necesarios. Estos pueden ser actualizables durante la ejecución del programa si el usuario lo ha programado.
- Pins: Los elementos pin son los círculos amarillos que representan las entradas y las salidas del nodo. Para añadir o quitar nodos se requiere de la lista de entradas y salidas del nodo, esta puede ser una lista vacía o con un numero n de pins.

Para generar estos nodos por código se sigue la plantilla:

```
def printer(val,self,*argss):
    #  Mostrar en texto el valor de entrada.
    dpg.set_value(self.custom_text,val)
    # Ese mismo valor se envia al pin de salida
    return val,


class Printer(NodeV2):
    Title = "Printer"
```

```
        def __init__(self):
            f = printer # Funcion declarada por el usuario
            inp_list = ["val"] # Lista a modificar por usuario
            out_list = ["val"] # Lista a modificar por usuario

            # se genera el nodo
            super().__init__(f, inp_list, out_list, self.Title)

            # APARTADO STATIC
            self.custom_text = dpg.add_text("",parent=self.static)

        # FUNCIONAMIENTO INTERNO CUSTOMIZADO
        def recollect_inputs_frombackawrd_nodes(self):
            # Recojemos los pins de entrada  [val]
            inp = super().recollect_inputs_frombackawrd_nodes()
            # Anadimos a la lista el valor self [val,self]
            inp.append(self)
            return inp
```

Si el nodo no requiere valores obtenidos por otros medios aparte de los pins, el código se simplifica como:

```
    def redirect(*argss):
        # return mismos elementos que pins de salida
        return argss

    class Redirect(NodeV2):
        Title = "redirect"
        def __init__(self):
            inp_list = ["val"]
            out_list = ["val"]
            super().__init__(redirect, inp_list, out_list, self.Title)
```

Por tanto, se declara:

- Función del nodo con las variables necesarias y return de una lista o tupla con los valores para los pins de salida.
- Titulo: nombre del nodo
- inp_list: lista de pins de entrada (dict):
  inp_list = ”F1”:”slider”:”width”:60,”F2”:”slider”:”width”:60. Esto generará un nodo con entradas F1 F2 con dos snipets slider que desaparecen si se conecta un nodo a una entrada.

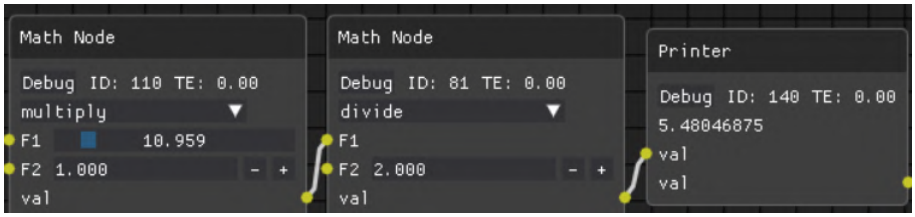


Figura 2.6: Entradas con cambio estético.

- out_pins: lista de pins de salida (string)
- En caso de requerir inputs no considerados por los pins se modifica la función $recollect\_inputs\_frombackawrd\_nodes(self)$.

2.4.1. Plantilla para generar nuevas librerías

```
from modules.father_class import NodeV2
import dearpygui.dearpygui as dpg

#######################################################
#######################################################
def exposition(val1,val2,*argss):
    return [val1**val2]

class Exposition(NodeV2):
    Title = "Exposition"
    def __init__(self):
        f = exposition
        inp_list = {"val":{"slider":{"width":120}},
                    "exp":{"value":{"width":120,"format": '%.8f'}}}
        out_list = ["val"]
        super().__init__(f, inp_list, out_list, self.Title)

        dpg.add_text("Static example",parent=self.static)

    def recollect_inputs_frombackawrd_nodes(self):
        inp = super().recollect_inputs_frombackawrd_nodes()
        # inp.append(dpg.get_value(self.type))
        return inp

    def custom_node_info(self):
        super().custom_node_info()
        dpg.add_text("Custom node info")
#######################################################
#######################################################

####
```

```
####
from modules.interaction import register
from modules import NODE_WINDOW_MENU
# REGISTRO DE NODOS
menu = "PLANTILLA"
with dpg.menu(label=menu, tag=menu, parent=NODE_WINDOW_MENU):
    pass

register_list = [
    Exposition,
    ]
for node in register_list:
    register(node, menu)
####
####
```

2.5. Funciones de la aplicación

- Procesado raw 2.7.
- Carga de imágenes de múltiples formatos 2.9.
- Gestión de color OCIO 2.10.
- Visualizado de imágenes.
- Guardado de imágenes.
- Facilidades para comprobar artefactos en perfiles de color 2.8.
- Facilidades para añadir Nodos nuevos por el usuario.

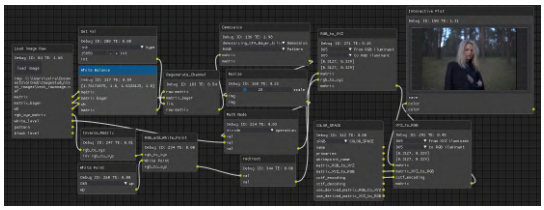


Figura 2.7: Procesado Raw

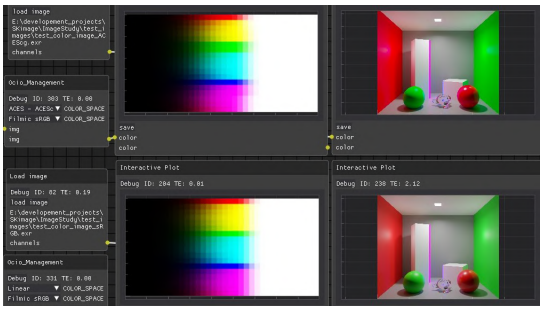


Figura 2.8: Facilidades para comprobar artefactos en perfiles de color

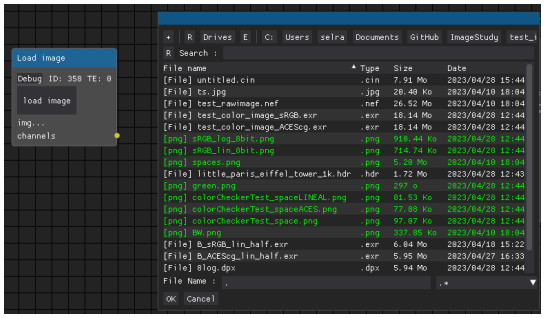


Figura 2.9: Formatos de imagen aceptados: JPG, png, hdr, tiff, exr, dpx etc...

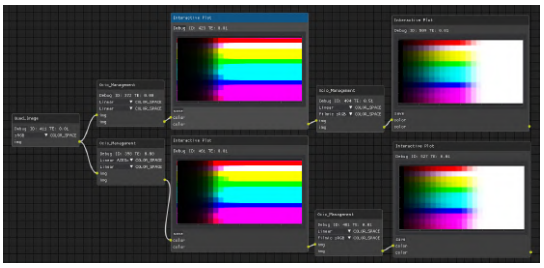


Figura 2.10: transformaciones mediante ocio

2.6. Ejemplo de uso

2.6.1. Generación de un LUT

Generación de un DRT desde cero 2.11. Algunos procesos como la generación de LUTs se simplifican con el uso de este programa. En el ejemplo 2.12 se observa como se genera un lut que podemos visualizar, la función que en este caso es de saturación y por ultimo el guardado de el archivo en su formato correspondiente.

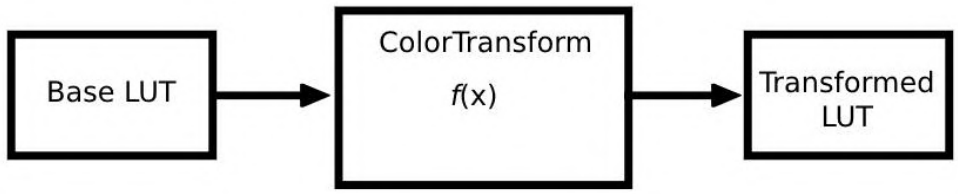


Figura 2.11: Flujo de trabajo

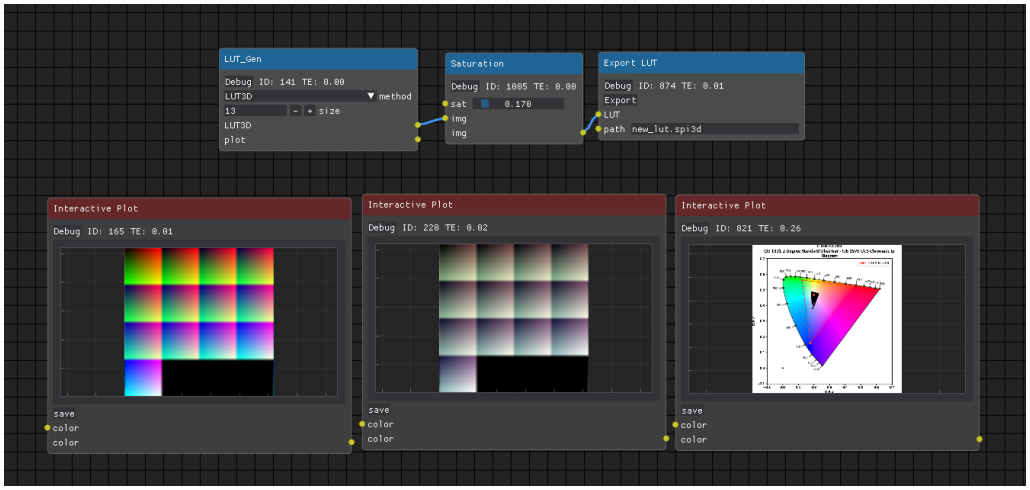


Figura 2.12: Ejemplo práctico

2.6.2. Visualización de características de las imágenes

Entre algunas de las funciones ya implementadas se encuentran:

1. Mostrar de forma segmentada las propiedades de exposición de las imágenes para un gran control sobre los colores2.13.

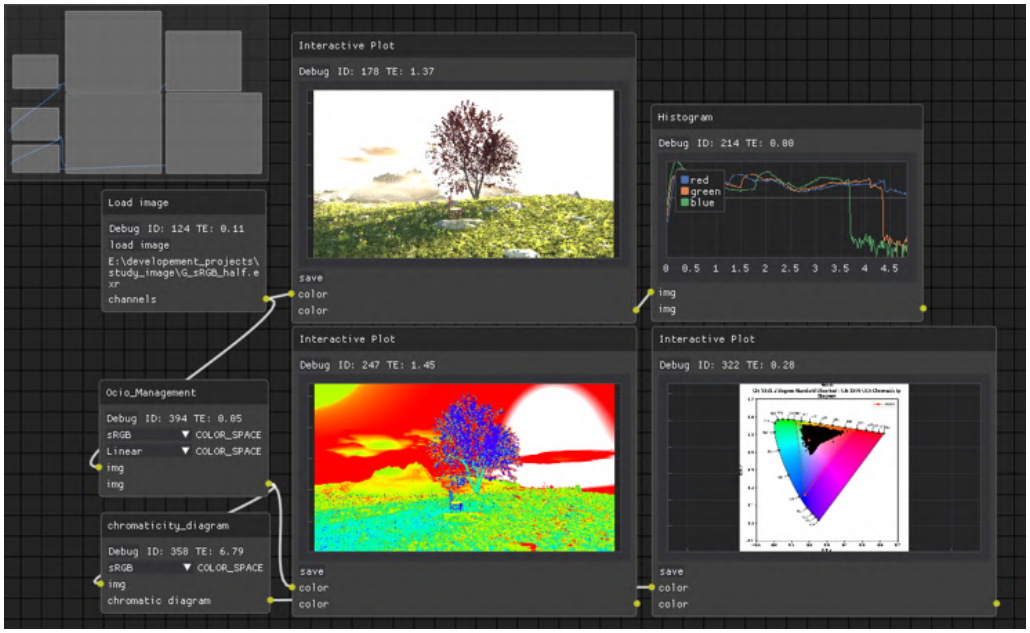


Figura 2.13: Exposición con técnica “False Color”

2. Comparar de forma rápida entre transformaciones de colores 2.14.

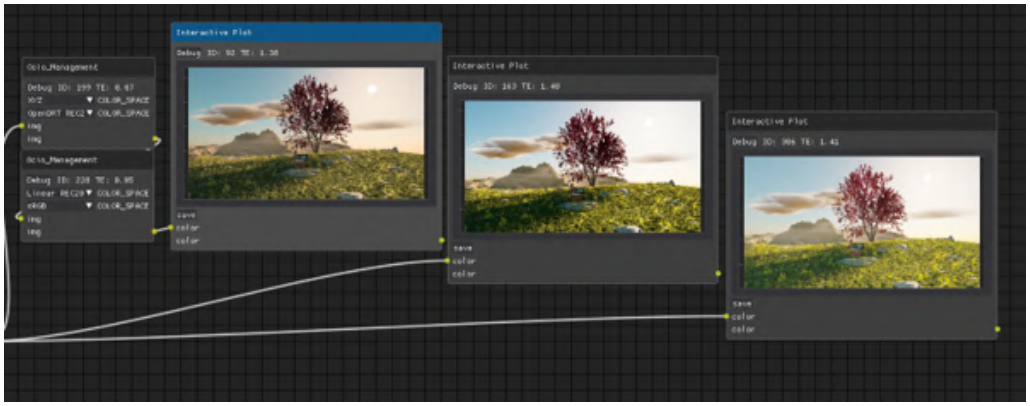


Figura 2.14: Comparación visual entre diferentes DRTs

Capítulo 3

Herramientas customizadas para software existente

3.1. Perfil Ocio Customizado

El programa desarrollado emplea una configuración ocio desarrollada específicamente para hacer experimentos. Con el uso de el propio programa se han creado los luts y funciones pertinentes para generar un nuevo DRT.

3.1.1. Funcionamiento de la configuracion OCIO

Todos los espacios de color se traducen a XYZ, su espacio PCS. De este modo, si una imagen entra en espacio ITU rec.2020 se puede transformar directamente a ACES con una transformación $ITUrec2020 -> default -> ACES$ que realizará la operación $ITUrec2020 -> XYZ -> ACES$.

3.1.2. Características únicas

A diferencia de las otras configuraciones, para esta configuración se busca un uso más experimental y con adaptación a diferentes entornos como blender o davinci. Por esta razón se trata de una configuración poco compleja. Además posee un nuevo DRT a diferencia de filmic, ACES, Nuke etc... Se trata de OpenDRT con dos modalidades, para espacio de color rec.709 y para rec.2020. El espacio de color para operaciones, como el renderizado o la corrección de color, es rec.2020 por su proximidad a un motor de renderizado espectral.

3.2. customOcio blender addon

En el software 3D blender, la gestión de color deja bastante que desear. A diferencia de otros programas, en blender solo se usa una configuración ocio con selección de múltiples espacios de entrada y un único espacio de salida.

Las nuevas funcionalidades que ofrece este addon son:

- Configuración de ocios personalizados: Si es necesario cambiar entre configuraciones se puede seleccionar entre todos los archivos que hayas situados en la configuración de blender. También incluye ocios internos que añaden soporte a los espacios ACEScg y ACES, AgX base, looks, Cineon Log y elección desde dentro del programa entre espacios de color para el renderizado (primarios sRGB y ACEScg).

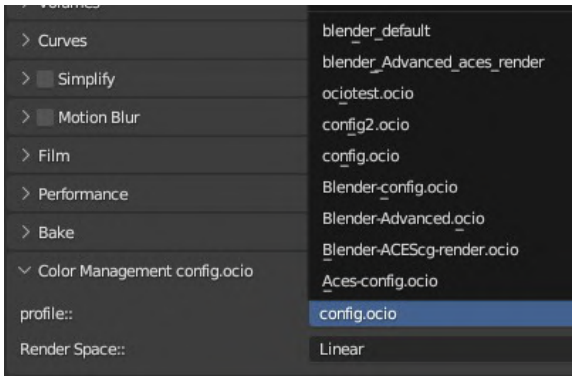


Figura 3.1: muestra del funcionamiento customConfig

- Nuevo config interno de blender (seleccionando blender_Advanced):
  - Transformaciones AgX y looks.
  - Transformaciones Filmic y looks.
  - OpenDRT experimental
  - Espacios ACES y ACES output.
  - Nueva organización de displays en [sRGB, XYZ, Utils].
  - Interoperabilidad de nomenclatura entre blender y ACES.

Bibliografía

[1] "The Principles of Color Management". En: *Digital Color Management: Principles and Strategies for the Standardized Print Production*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, págs. 58-75. ISBN: 978-3-540-69377-2. DOI: 10.1007/978-3-540-69377-2_3. URL: https://doi.org/10.1007/978-3-540-69377-2_3.

[2] *Light Objects*. Jun. de 2023. URL: https://docs.blender.org/manual/en/latest/render/lights/light_object.html.

[3] Snehasish Dutta Gupta y Avinash Agarwal. *Artificial Lighting System for Plant Growth and Development: Chronological Advancement, Working Principles, and Comparative Assessment*. Oct. de 2017, págs. 1-25. ISBN: 978-981-10-5806-6. DOI: 10.1007/978-981-10-5807-3_1.

[4] CIE (Commission Internationale de l'Eclairage). *Colorimetry - 4th Edition*. Vienna, Austria: CIE Central Bureau, 2018.

[5] Chris Brejon. *Ocio, display transforms and misconceptions*. Sep. de 2021. URL: https://chrisbrejon.com/articles/ocio-display-transforms-and-misconceptions/.

[6] Richard Butler. *Raw bit depth*. Richard Butler, 2017. URL: https://www.dpreview.com/articles/4653441881/bit-depth-is-about-dynamic-range-not-the-number-of-colors-you-get-to-capture.

[7] URL: https://opencolorio.org/.

[8] Microsoft. *Bitmap (BMP) Files*. Referenced September 2021. URL: https://learn.microsoft.com/en-us/windows/win32/gdi/bitmaps.

[9] PNG Development Group. *PNG (Portable Network Graphics) Specification, Version 1.2*. http://www.libpng.org/pub/png/pngintro.html. Accedido el 17 de mayo de 2023. 1996.

[10] S. Naveen Kumar, M. V. Vamshi Bharadwaj y Shreyanka Subbarayappa. "Performance Comparison of Jpeg, Jpeg XT, Jpeg LS, Jpeg 2000, Jpeg XR, HEVC, EVC and VVC for Images". En: *2021 6th International Conference for Convergence in Technology (I2CT)*. 2021, págs. 1-8. DOI: 10.1109/I2CT51068.2021.9418160.

[11] Truevision, Inc. *Truevision TGA: FILE FORMAT SPECIFICATION VERSION 2.0*. Technical Manual Version 2.2. Truevision, Inc. 7340 Shadeland Station, Indianapolis, IN 46256-3925: Truevision, Inc., ene. de 1991. URL: https://www.dca.fee.unicamp.br/~martino/disciplinas/ea978/tgaffs.pdf.

[12] Richard Patterson. *Understanding Cineon*. First Draft 10/2/01. 6700 Valjean Avenue, Van Nuys, CA 91406: Illusion Arts, oct. de 2001. URL: http://www.digital-intermediate.co.uk/film/pdf/Cineon.pdf.

[13] "ST 268:2003 - SMPTE Standard - For File Format for Digital Moving-Picture Exchange (DPX), Version 2.0". En: *ST 268:2003* (2003), págs. 1-17. DOI: 10.5594/SMPTE.ST268.2003.

[14] URL: https://openexr.com/en/latest/TechnicalIntroduction.html.

[15] Ricky Reusser. *Half-precision floating-point, visualized*. Mar. de 2021. URL: https://observablehq.com/@rreusser/half-precision-floating-point-visualized.

[16] Julien Guertault y Julien Guertault. *RGBD Archives*. Mayo de 2013. URL: http://lousodrome.net/blog/light/tag/rgbd/.

[17] Adobe Developers Association. *TIFF*. Revision 6.0. 1585 Charleston Road, P.O. Box 7900, Mountain View, CA 94039-7900: Adobe Systems Incorporated, jun. de 1992. URL: http://www.adobe.com/Support/TechNotes.html.

[18] URL: http://dotwhat.net/file/extension/webp/10843.

[19] Scott Miller. "2018 Update on High Dynamic Range Television". En: *SMPTE Motion Imaging Journal* 127.8 (2018), págs. 91-93. DOI: 10.5594/JMI.2018.2849659.

[20] Anders Langlands. *Anders Langlands*. URL: https://www.colour-science.org/anders-langlands/.

[21] Jed Smith. *Jedypod/open-display-transform: Open display transform is a collection of tools and experiments for rendering wide-gamut scene-linear data into an image for an SDR or HDR display device.* URL: https://github.com/jedypod/open-display-transform.

Parte II

Anexos

Apéndice A

Listados adicionales

A.1. Datos obtenidos de las investigaciones.

A.1.1. Ejemplos de diferentes espacios de color

- **Human perception**
  - CIE 1931 XYZ
  - CIEUVW
  - Uniform color spaces
    - CIELUV
    - CIELAB
    - HSLuv
- **RGB primaries**
  - sRGB
  - Adobe RGB
  - Adobe Wide Gamut RGB
  - Rec. 2100
  - ProPhoto RGB color space
  - scRGB
  - DCI-P3, used primarily for digital movie projection
  - SMPTE 240M / SMPTE Ç”, used in NTSC and MUSE analog television systems
  - Rec. 601, used for SDTV
  - Rec. 709, used for HDTV
  - Rec. 2020, used for UHDTV
  - Academy Color Encoding System (ACES)
    - ACES2065-1
    - ACEScg

    - ACEScc & ACEScct
    - ACESproxy
  - ...

- **YCbCr and YUV**
  - YPbPr
  - YDbDr
  - YIQ
  - xvYCC
  - sYCC

- **Cylindrical transformations**
  - HSV and HSL
  - LCh: uniform color space

- **Subtractive**
  - CMYK and CMY

- **Commercial color spaces**
  - Munsell color system – early perceptually-uniform color space
  - Natural Color System (NCS) – perceptual
  - Pantone Matching System (PMS) – standardized color reproduction (and color list)
  - RAL – standardized color matching (and color list)
  - Aerospace Material Specification - Standard 595A (Supersedes (US) Federal Standard 595C)[12]
  - (US) Federal Standard 595C
  - British Standard Colour (BS)
  - HKS – standardized color reproduction (and color list)

- **Special-purpose color spaces**
  - The rg chromaticity space is used in computer vision applications, and shows the color of light (red, yellow, green, etc.), but not its intensity (dark, bright).
  - LMS color space (long, medium, short), a perceptual color space based on the response functions of the cones in the retina of the eye. It is mostly used in psychophysical research.
  - TSL color space is used in face and skin detection.

- **Obsolete color spaces**
  - RG for early Technicolor film
  - RGK for early color printing

A.1.2. Primarios de múltiples espacios de colores

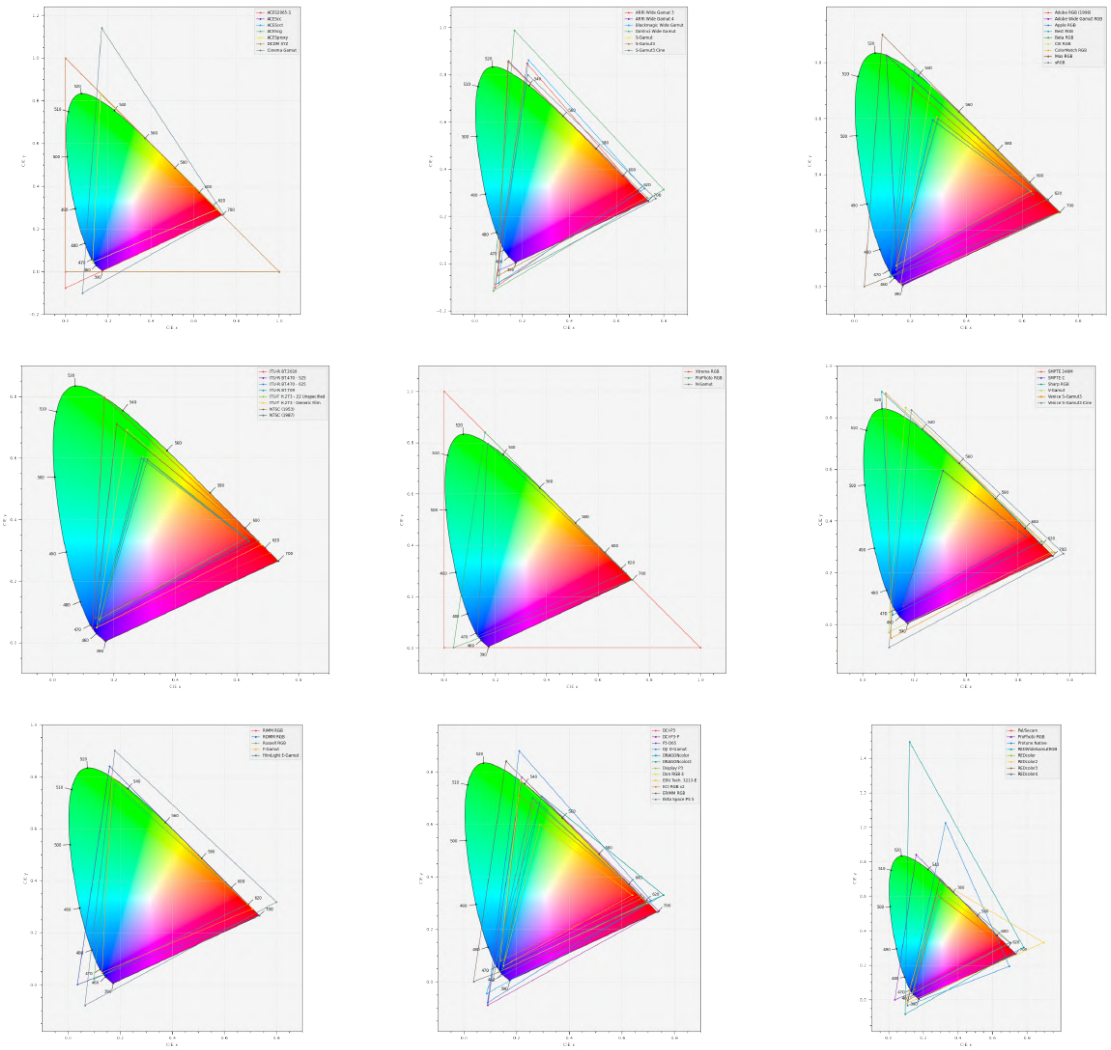


Figura A.1: Primarios RGB de múltiples espacios de colores

A.1.3. Comparación de peso entre formatos de imagen

Formato	Peso
bmp	1.87 GB
Cineon	1.24 GB
DPX 10bit	1.25 GB
DPX 12bit	1.87 GB
DPX 16bit	1.87 GB
DPX 8bit	934.33 MB
JPEG	74.64 MB
png	489.92 MB
png 16bit	1.30 GB
Radiance HDR	835.83 MB
Targa	854.26 MB
Targa Raw	933.12 MB
TIFF 16bit	1.76 GB
TIFF 8bit	558.81 MB
WebP	63.02 MB
JPEG2000 12bit J2K	116.62 MB
JPEG2000 12bit JP2	116.62 MB
JPEG2000 16bit J2K	155.50 MB
JPEG2000 16bit JP2	155.50 MB
JPEG2000 8bit J2K	77.74 MB
JPEG2000 8bit JP2	77.75 MB

Tabla A.1: Tamaño de formatos. 150 frames 1920x1080

Formato	Peso
OpenEXR FULL B44	3.73 GB
OpenEXR FULL B44A	3.73 GB
OpenEXR FULL DWAA	411.75 MB
OpenEXR FULL DWAB	396.66 MB
OpenEXR FULL PIZ	3.10 GB
OpenEXR FULL Pxr24	1.63 GB
OpenEXR FULL RLE	3.74 GB
OpenEXR FULL ZIP	3.52 GB
OpenEXR FULL ZIPS	3.68 GB
OpenEXR HALF B44	816.61 MB
OpenEXR HALF B44A	816.15 MB
OpenEXR HALF DWAA	411.75 MB
OpenEXR HALF DWAB	396.66 MB
OpenEXR HALF PIZ	1.01 GB
OpenEXR HALF Pxr24	977.56 MB
OpenEXR HALF RLE	1.39 GB
OpenEXR HALF ZIP	944.87 MB
OpenEXR HALF ZIPS	1.00 GB
OpenEXR MultiLayer FULL B44	4.98 GB
OpenEXR MultiLayer FULL B44A	4.98 GB
OpenEXR MultiLayer FULL DWAA	411.96 MB
OpenEXR MultiLayer FULL DWAB	396.73 MB
OpenEXR MultiLayer FULL PIZ	3.24 GB
OpenEXR MultiLayer FULL Pxr24	1.64 GB
OpenEXR MultiLayer FULL RLE	4.40 GB
OpenEXR MultiLayer FULL ZIP	3.57 GB
OpenEXR MultiLayer FULL ZIPS	3.69 GB
OpenEXR MultiLayer HALF B44	1.09 GB
OpenEXR MultiLayer HALF B44A	874.50 MB
OpenEXR MultiLayer HALF DWAA	411.92 MB
OpenEXR MultiLayer HALF DWAB	396.71 MB
OpenEXR MultiLayer HALF PIZ	1.05 GB
OpenEXR MultiLayer HALF Pxr24	987.37 MB
OpenEXR MultiLayer HALF RLE	1.40 GB
OpenEXR MultiLayer HALF ZIP	955.25 MB
OpenEXR MultiLayer HALF ZIPS	1.01 GB

Tabla A.2: Tamaño de EXR. 150 frames 1920x1080