UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Aportaciones a la reconstrucción de la reflectancia espectral de una carta de color mediante la captura de imágenes con cámara digital tricromática y distintos iluminantes

Doctorando: José Vicente del Valle Fayos

Directores: Fernando Brusola Simón
Ignacio Tortajada Montañana

Valencia, enero de 2015

Cuando las gotas de rocío discurren por las hojas de nuestra hierba fresca
siento el viento frío escocer mi tez y mis ojos lloran de ver amanecer
y las lágrimas me hacen ver nuestra tierra tan hermosa...
aquella con olor a chocolate y sabor a café...

— José Vicente del Valle Fayos, enero de 2009

Agradecimientos

Empecé mis estudios universitarios en el año 1999. Aquel año y los siguientes fueron un poco complicados para mí, tanto en lo personal como en lo académico. Afortunadamente, el entonces Rector de la UPV, Justo Nieto, me dió un voto de confianza y me concedió un año adicional de permanencia en la universidad. Mis padres, por su parte, estuvieron ahí e hicieron las cosas lo mejor que pudieron. Gracias a Justo y gracias a mis padres.

Al año siguiente obtuve los créditos que se exigen en las condiciones de permanencia de la UPV. Durante ese curso acudí a la psicopedagoga del Instituto de Ciencias de la Educación de la UPV, Pilar Bonet. Ella me orientó y me ayudó. Gracias a Pilar.

Durante el curso 2001/02 y durante los cursos siguientes fui capaz de aprobar todas las asignaturas.

Durante mis estudios en la Escuela Técnica Superior de Ingeniería del Diseño y en la Escuela Técnica Superior de Ingenieros Industriales tuve buenos profesores y buenos amigos entre los cuales se encuentran mis amigos del PIE y otros como Maite García, Luís de Torres, Laura Solaz, Clara Morant, Aitor Valero y Maribel Ramírez.

En el año 2008 me matriculo oficialmente de doctorado. Mi relación con Fernando Brusola –mi director de tesis– tuvo sus altibajos, como suele ocurrir en la mayor parte de relaciones estrechas. Fernando Brusola despertó en mi el interés por la ciencia del color y dirigió mis PFCs. Gracias a Fernando.

En el año 2010 todavía persisten algunos de los problemas que venía arrastrando desde el año 1999, pero hoy puedo decir que estoy viviendo una de las mejores etapas de mi vida. Me encuentro bien conmigo mismo y con los que me rodean. En este sentido quiero mostrar mi agradecimiento a Luis Rojo.

Con la entrada en vigor de la nueva normativa de los estudios de doctorado me veo obligado a terminar antes de febrero de 2016. Con preocupación acudo al director de la ETSID, Enrique Ballester, quien me asigna un nuevo director de tesis, Ignacio Tortajada. Ignacio es muy amable, me dedica su tiempo, me anima y me orienta. Gracias a Enrique y gracias a Ignacio hoy va a ser presentada esta tesis.

No puedo acabar esta sección sin agradecer a Vicent Giner que me haya facilitado la plantilla de LaTeXy a Miguel Alfonso Mollar que nos haya prestado su espectrofotómetro para medir la curva espectral de los iluminantes.

Finalmente quiero agradecer a mi familia, en especial a mis padres y a mis hermanas, el apoyo incondicional prestado a lo largo de todos estos años.

Gracias por estar aquí. Gracias a tod@s.

Resumen

Con el propósito de evitar el efecto del metamerismo en la medición de color de los dispositivos de captura de imágenes, se han desarrollado técnicas basadas en el uso de filtros de banda ancha o estrecha acoplados a una cámara digital monocromática de laboratorio para capturar información de color a distintas longitudes de onda y poder realizar la reconstrucción de la reflectancia espectral de la escena basada en un análisis de componentes principales, un análisis de componentes independientes o una matriz de pseudoinversa directa.

Lo que no ha quedado cubierto por el estado del arte es la posibilidad de emplear una técnica de reconstrucción basada, no en el uso de filtros, sino en el uso de iluminantes con diferentes temperaturas de color y una cámara doméstica.

En el presente trabajo se explora esa posibilidad y se proponen además dos métodos adicionales para la reconstrucción de la reflectancia espectral, basados en el cálculo de las mínimas distancias Euclídeas dentro del espacio Lab entre el parche de color que se desea reconstruir y un subconjunto de parches de color de la carta de entrenamiento [1].

[1] Hunter et al. [1] presentaron un escáner de dos iluminantes destinado a obtener más precisión en la medición del color, reducir el ruido de la imagen y corregir errores en la lectura de superficies con relieve. El presente trabajo sigue esa línea, pero su objeto es el de reconstruir la curva de reflectancia espectral dadas las lecturas tomadas por una cámara fotográfica bajo dos iluminantes con distintas temperaturas de color y comparar los resultados empleados al utilizar distintos algoritmos de reconstrucción.

Abstract

With the purpose of avoiding the effect of metamerism in the color measurement of imaging devices, it has been developed some techniques based on the use of broadband filters or narrow filters coupled to a monochrome digital laboratory camera to capture color information at different wavelengths and perform the reconstruction of the spectral reflectance of the scene based on a principal component analysis, independent analysis components or direct pseudoinverse matrix.

What has not been covered by the state of the art is the possibility of using a reconstruction technique based not on the use of filters, but using illuminants with different color temperatures and a home camera.

In the present paper we explore this possibility and further propose two additional methods for the reconstruction of spectral reflectance, based on the calculation of the minimum Euclidean distances in Lab space between the color patch to be reconstructed and a subset patch color chart workout [2].

[2]Hunter et al. [1] presented two illuminating scanner aimed at more accurate color measurement, reduce image noise and correct errors in reading embossed surfaces. This paper follows this line, but the purpose is to reconstruct the spectral reflectance curve given readings taken by a camera under two illuminants with different color temperatures and compare the results using different reconstruction algorithms.

Resum

Amb el propòsit d'evitar l'efecte del metamerism en la medició de color dels dispositius de captura d'imatges, s'han desenvolupat tècniques basades en l'ús de filtres de banda ampla o estreta acoplats a una càmera digital monocromàtica de laboratori per capturar informació de color a diferents longituds d'ona i poder realitzar la reconstrucció de la reflectància espectral de l'escena basada en una anàlisi de components principals, una anàlisi de componenetes independents o una matriu de pseudoinversa directa.

El que no ha quedat cobert per l'estat de l'art és la possibilitat de utilitzar una tècnica de reconstrucció basada, no en l'ús de filtres, sinó en l'ús de iluminants amb diferents temperatures de color i una càmera domèstica.

En el present treball de recerca s'explora aquesta possibilitat i es proposen a més dos mètodes addicionals per a la reconstrucció de la reflectància espectral, basats en el càlcul de les mínimes distàncies Euclidianes dins de l'espai Lab entre la mostra de color que es desitja reconstruir i un subconjunt de mostres de color de la carta d'entrenament[3].

[3]Hunter et al. [1] van presentar un escàner de dos iluminants destinat a obtenir més precisió en el mesurament del color, reduir el soroll de la imatge i corregir errors en la lectura de superfícies amb relleu. Aquest treball segueix aquesta línia, pero el seu objecte és el de reconstruir la corba de reflectància espectral donades les lectures preses per una càmera fotogràfica sota dos iluminants amb diferents temperatures de color i comparar els resultats empleats en utilitzar diferents algoritmes de reconstrucció.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En el presente capítulo se justifica la necesidad de desarrollar sistemas de captura espectral de imágenes, se propone una clasificación de estos sistemas, se realiza una revisión bibliográfica extensa sobre los métodos (algoritmos) empleados para la reconstrucción espectral de imágenes algunos de los cuales serán empleados en los apartados siguientes y para finalizar se clasifica brevemente el ruido producido por estos sistemas y se enumeran las distintas técnicas que se pueden emplear para minimizar su efecto. Al final de esta parte dedicada al ruido dentro del capítulo 1 se expone de qué forma se aborda su problemática en el presente trabajo.

1.1. Justificación y alcance

En la actualidad los dispositivos de captura de imágenes representan la información visual en un conjunto ordenado de elementos llamados píxeles, cada uno de los cuales queda definido con una terna de valores RGB [2], [3].

Pero la información colorimétrica completa de una muestra de color queda definida por su reflectancia espectral o por su emitancia espectral[1] no por sus valores tricromáticos, ya que una combinación particular de iluminante, muestra y observador puede dar lugar a un estímulo visual coincidente para dos muestras de color que dejan de parecer iguales cuando cambia el iluminante, el observador o ambos. A este fenómeno se le conoce con el nombre de metamerismo [4] y es la causa por la cual los dispositivos de captura tricromática de imágenes ofrecen mediciones del color dependientes de las condiciones de observación.

Con objeto de medir la reflectancia espectral en lugar de los valores tricromáticos y para evitar el metamerismo se han desarrollado dispositivos que tienen el inconveniente de medir el color en un único punto, además de ser caros [5], [6]. También se han desarrollado dispositivos capaces de medir la reflectancia espectral de millones de puntos de color simultáneamente basados en sucesivos filtros de banda ancha o estrecha que se acoplan a una cámara monocromática. Y además se ha desarrollado un escáner capaz de reducir el error en la medida del color empleando dos iluminantes distintos.

El presente trabajo de investigación sigue la línea propuesta por Hunter et al. [1] y también [7], [8], [9], [10], [11], [12], [13], [14], pero su objeto es el de comparar la bondad de las reconstrucciones espectrales empleando distintos métodos, algunos ya propuestos en la literatura y otros como 2.5 y 2.6 propuestos aquí, y haciendo uso de una cámara doméstica. El método de los colores cercanos y el método cuadrático de los colores cercanos están basados en una idea de la tesis de Plata [15] en la que se toma una esfera de un radio determinado a priori en

[1] Según refleje o emita luz.

el espacio Lab y se toma como referencia para la reconstrucción todos los colores contenidos en esa esfera. La diferencia entre esta forma de proceder de Plata y la forma de proceder en el presente trabajo consiste en que aquí se toma un valor n de parches más cercanos al parche que se desea reconstruir en lugar de tomar el radio de una esfera.

1.2. Clasificación de sistemas de captura espectral

Los sistemas de captura espectral de color pueden clasificarse atendiendo al número de canales de entrada del sistema o a la técnica empleada para capturar la escena.

1.2.1. Atendiendo al número de canales de entrada

En función del número de canales de entrada [16], [17], nos encontramos con distintos tipos de sistemas de adquisición espectral de imágenes:

- 1 canal: Monocromático
- 3 canales: Tricromático
- 4–9 canales: Multiespectral [18]
- 10–100 canales: Hiperespectral [19]
- $>$100 canales: Ultraespectral

1.2.2. Atendiendo a la técnica empleada para capturar la escena

Las técnicas empleadas para capturar la información espectral de la escena estan basadas en el uso de filtros que pueden ser de banda ancha o de banda estrecha así como en el uso de distintos iluminantes (Imai et al. [20], Sherstha et al. [21], Hernandez-Andrés et al. [22], Cheung et al. [23], Hardeberg et al. [24]).

Uso de filtros de banda ancha

Este tipo de sistemas ha sido ampliamente utilizado por Imai et al.[25], [20] y Plata [26]. Cada imagen tomada con un nuevo filtro añade tres canales de color al sistema global. El número y forma de los filtros de banda ancha que se deben emplear para que la estimación espectral sea óptima dependen de la forma espectral de los propios canales RGB de la cámara, del ruido presente en la misma y de la forma de los espectros que se quieran reconstruir (Imai et al. [20]). Para poder utilizar estos sistemas es imprescindible llevar a cabo un entrenamiento previo de los mismos. Básicamente el entrenamiento consiste en relacionar las respuestas de la cámara a estímulos espectrales conocidos, para poder así construir una matriz que será después necesaria para aplicar algún algoritmo de estimación espectral [27], [28].

Uso de filtros de banda estrecha

Los filtros interferenciales (o los sintonizables, como el de cristal líquido LCFT utilizado por Brettel et al. [29] o Hardeberg et al. [30]) son de banda estrecha y normalmente de perfil espectral gaussiano, lo que permite utilizarlos de dos formas diferentes a la hora de tratar de conseguir medidas de curvas espectrales, [31], pero siempre en conjunción con cámaras monocromas. La primera de ellas consiste en utilizar un número reducido de canales o filtros y algún algoritmo de estimación espectral del mismo modo que se hace con los sistemas de banda ancha, por lo que de esta manera se requiere un entrenamiento previo del sistema. La segunda forma consiste en suponer que el filtro es completamente monocromático y que, por tanto, sólo proporciona información de aquella longitud de onda en la que dicho filtro presenta su máxima transmitancia [32], [33]. Si la cámara con la que se utilizan estos filtros está calibrada radiométricamente y si se conoce la transmitancia del filtro, podemos muestrear directamente en cada longitud de onda la radiancia incidente sobre el sistema. Si se desea obtener mayor resolución espectral que la dada por las posiciones de máximo de los filtros, siempre se pueden

utilizar técnicas de interpolación o un número mayor de filtros (sistema hiperespectral [34], [35]). Los sistemas multiespectrales o hiperespectrales de banda estrecha basados en muestreo no requieren pues la utilización de algoritmos de estimación espectral, ni entrenamiento previo del sistema.

Uso de distintos iluminantes

Consiste en tomar dos instantáneas de la escena con ayuda de una cámara doméstica y dos iluminantes[2] en las mismas condiciones de distancia focal, apertura, tiempo de exposición, ISO, balance de blancos, etc...de modo que así se obtienen 6 canales de los 4 que se exigen para realizar una reconstrucción espectral aceptable según Shi et al. [36] para posteriormente aplicar algun método basado en el cálculo de la relación existente entre la información proporcionada por esos canales y la información medida con un espectrofotómetro. Conocida esta relación procedente de un conjunto de parches de color de entrenamiento, es posible predecir la reflectancia de un conjunto de parches de test con solo medir sus valores tricromáticos bajo cada uno de los iluminantes [37].

1.3. Métodos para la reconstrucción espectral

Una curva de reflectancia espectral en el espectro visible (entre 380 y 730 nm) puede quedar determinada a partir de un conjunto de muestras tomadas a intervalos de 10nm. Dichas curvas pueden ser entonces tratadas matemáticamente como vectores en un espacio N–dimensional, con $N = 36$.

El objetivo de un sistema multiespectral es tratar de estimar esas N muestras con la mayor exactitud posible, utilizando para ello la información obtenida a partir de un número k reducido de sensores, mucho menor que el número N de muestras [38].

[2]Cuanto más difieran espectralmente estos iluminantes, mejor reconstrucción se espera.

Cuando se utiliza una cámara para capturar una escena con reflectancia espectral R^x, la respuesta del sensor k–ésimo en un píxel x puede expresarse según:

$$q_k^x = \int_{\lambda_1}^{\lambda_2} R^x(\lambda) E(\lambda) Q_k(\lambda)\, \mathrm{d}\lambda \tag{1.1}$$

Donde $R^x(\lambda)$ es la reflectancia espectral en función de λ en el píxel x, $E(\lambda)$ es la distribución de potencia espectral del iluminante (SPD, spectral power distribution) y $Q_k(\lambda)$ es la sensibilidad espectral del sensor para el canal k–ésimo.

Si suponemos que el sensor proporciona una respuesta lineal, podemos expresar la reflectancia espectral como una suma ponderada de funciones base $R_j(\lambda)$:

$$R^x(\lambda) = \sum_{j=1}^{n} \sigma_j^x R_j(\lambda) \tag{1.2}$$

Donde el número de elementos de la base, n, se correponde con los grados de libertad del modelo. Los valores buscados son los pesos σ_j^x del desarrollo en serie.

Algo similar puede hacerse para describir la SPD del iluminante:

$$E(\lambda) = \sum_{i=1}^{m} \epsilon_i E_i(\lambda) \tag{1.3}$$

De modo que en este caso las bases serían los elementos $E_i(\lambda)$. Los m valores de ϵ_i en la ecuación (1.3) forman un vector columna ϵ que especifica el iluminante ϵ_i, y los n valores σ_j forman un vector columna σ que proporciona la reflectancia de la superficie $R^x(\lambda)$. Con esto, podemos expresar la reflectancia de una escena de forma matricial según:

$$[R] = [V][\sigma] \tag{1.4}$$

donde $[R]$ es una matriz $N \times x$, $[V]$ es la matriz $N \times n$ combinación lineal de la base de vectores y $[\sigma]$ es la matriz de coeficientes $N \times x$.

Si sustituimos las ecuaciones (1.2), (1.3) y (1.4) en la ecuación (1.1), podemos expresar la relación entre las respuestas de la cámara y la reflectancia de la superficie a través de la ecuación matricial:

$$[q] = [E_Q]^T[R] \tag{1.5}$$

donde las respuestas de la cámara en cada píxel de la imagen, $[q]$, vienen expresadas por la matriz $k \times x$ siendo k el número de sensores; $[E_Q]$ es la matriz[3] $N \times k$ resultante de multiplicar longitud de onda a longitud de onda las componentes del vector iluminante $E(\lambda)$ y la matriz de sensibilidades de la cámara $Q_k(\lambda)$ y $[R]$ es, de nuevo, la matriz $N \times x$ que contiene las reflectancias en cada punto de la superficie.

Por otra parte, cualquier dispositivo multiespectral real está afectado por diversas fuentes de ruido (dcnoise, ruido térmico, shot noise, ruido flicker y ruido de cuantización). Si asumimos que la respuesta del sensor $[q]$ es lineal, la respuesta del sensor en presencia de ruido, $[q']$, puede expresarse en función de la respuesta sin ruido como:

$$[q'] = [q] + [\eta] \tag{1.6}$$

donde $[\eta]$ es una matriz $k \times x$ de componentes independientes aleatorias y no correlacionadas que afecta a cada sensor de forma independiente. El término de ruido $[\eta]$ contiene también otros factores que modifican la respuesta teórica del sistema, como pueden ser errores experimentales o de calibración de los componentes del mismo.

Como puede verse en la ecuación (1.5), la estimación espectral de $[\sigma]$ a partir de las respuestas de unos pocos sensores es un problema indeterminado. Dado que hay más incógnitas (longitudes de onda) que ecuaciones (sensores), existen infinitas curvas espectrales que pueden dar lugar a las mismas respuestas de los sensores y, por tanto, serían indistinguibles para el sistema multiespectral (dando lugar a lo que se conoce como metamerismo de la cámara).

La mayoría de los algoritmos de estimación espectral tratan de

[3] Traspuesta en la ecuación y denotada por el superíndice T.

resolver la anterior indeterminación siguiendo criterios de mejor solución promedio (mínimos cuadrados o minimización de alguna métrica, etc...) en algún subespacio englobado dentro del espacio de vectores que representan las curvas espectrales. Dichos algoritmos se basan en el conocimiento *a priori* del tipo de espectro que se desea recuperar o estimar, para así aportar soluciones con características espectrales similares a la solución deseada. Es frecuente utilizar algún método de extracción de información como el análisis de componentes principales (PCA, Principal Component Analysis), o, más recientemente la factorización no–negativa de matrices (NMF, Non–negative Matrix Factorization) o el análisis de componentes independientes (ICA, Independent Component Analysis).

Estos métodos se caracterizan por proporcionar un conjunto de vectores base a partir de una serie de espectros de entrenamiento cuyas características espectrales son similares a las de aquellos espectros que queremos recuperar [39]. En el caso del ICA los vectores base son independientes pero no necesariamente ortonormales ni con una estadística marcada según el orden del vector, mientras que en el NMF los vectores base son independientes y siempre positivos.

En los siguientes apartados se presentan varios métodos de estimación espectral.

1.3.1. Algoritmo de Maloney–Wandell

El método propuesto por Maloney et al. [40] parte de la ecuación (1.5), despreciando el efecto del ruido:

$$[q] = [E_Q]^T[R] = [E_Q]^T[V][\sigma] = [\Lambda_\epsilon][\sigma] \tag{1.7}$$

Donde la matriz $[\Lambda_\epsilon]$ es una matriz $k \times n$ cuya k–ésima entrada tiene la forma $\int R_j(\lambda)E(\lambda)Q_k(\lambda)\,\mathrm{d}\lambda$ y que relaciona directamente los pesos σ de la combinación lineal con la respuesta q de los sensores [41], [42]. Dicha matriz se obtiene a partir del conocimiento de las sensibilidades espectrales de la cámara y de la base de vectores representativos de los espectros de entrenamiento que se obtienen a partir de éstos, como hemos

dicho, mediante PCA, ICA o NMF:

$$[\Lambda_\epsilon] = [E_Q]^T[V] \tag{1.8}$$

Una vez calculada $[\Lambda_\epsilon]$, podemos obtener los coeficientes σ para la estimación espectral a partir de las respuestas de los sensores de una curva SPD desconocida, de acuerdo con la expresión[4]:

$$[\sigma] = [\Lambda_\epsilon]^+[q] \tag{1.9}$$

Nótese que en este método es necesario conocer las sensibilidades espectrales de los sensores del sistema y no se hace uso alguno de la estimación del ruido, lo que hace que este algoritmo sea muy poco robusto frente a dicho ruido [43], [44] y que cualquier pequeño error en la estimación de los sensores se traduzca en un error en la recuperación del espectro R [45].

1.3.2. Algoritmo de Imai–Berns

El método propuesto por Imai et al. [25] busca una relación directa entre las respuestas q de los sensores y los coeficientes de la combinación lineal σ. Si tenemos un conjunto de m vectores de entrenamiento a los que designaremos con el subíndice ts (training set), podemos relacionar ambos parámetros según:

$$[\sigma_{ts}] = [A][q_{ts}] \tag{1.10}$$

Siendo $[A]$ una matriz $n \times k$ que es formalmente parecida a $[\Lambda_\epsilon]$ del método de Maloney–Wandell, pero que se calcula de forma diferente ya que en esta ocasión también necesitamos información sobre las respuestas de los sensores a los espectros de entrenamiento q_{ts}. Como estas respuestas están afectadas por ruido, al utilizar la matriz $[A]$ ya estamos teniendo en cuenta, de alguna forma, el ruido presente en el sistema. La matriz $[A]$ se calcula por pseudoinversión:

[4] El signo $+$ indica que se ha calculado la pseudoinversa de Moore-Penrose.

$$[A] = [\sigma_{ts}][q_{ts}]^{+} \tag{1.11}$$

Una vez obtenida $[A]$, podemos recuperar el espectro deseado a partir de las respuestas que los sensores registran de dicho espectro:

$$[R] = [V][A][q] \tag{1.12}$$

De esta forma, la información sobre los espectros de entrenamiento se encuentra tanto en $[V]$ como en $[A]$, que también contiene información del ruido presente en el sistema, por lo que este método será robusto frente al ruido. Para probar esa afirmación no hay más que ver, a partir de las ecuaciones (1.7) y (1.10) que:

$$[A] = [\sigma_{ts}] \left([E_Q]^T [R_{TS}] + [\eta] \right)^{+} \tag{1.13}$$

Al igual que con el método de Maloney–Wandell, la calidad de la base $[V]$ es crucial para obtener buenas recuperaciones espectrales.

1.3.3. Algoritmo de Shi–Healey

Los algoritmos de Maloney et al. [40] y de Imai et al.[25] han sido utilizados en multitud de trabajos de recuperación espectral de reflectancias de todo tipo de objetos e iluminantes. Un resultado común a todos ellos se refiere al número óptimo n de vectores base utilizados para la recuperación de los espectros, que en todos los casos resulta ser igual al número k de sensores utilizados en el sistema. Pese a que ambos algoritmos permiten la utilización de un número de vectores mayor que el de sensores [46], [47], [48] ello implica tener que aplicar operaciones de pseudoinversión a la hora de operar con las matrices $[\Lambda_\epsilon]$ y $[A]$, en lugar de poder utilizar inversas sencillas con solución unívoca. El resultado de utilizar la pseudoinversa de Moore-Penrose no es más que la solución de mínimos cuadrados a un problema con más incognitas que ecuaciones que, por tanto, tiene infinitas soluciones posibles. Esta ambigüedad en el conjunto de soluciones posibles a la hora de resolver estos algoritmos

con $n > k$, unido al hecho de que el ruido presente en el sistema afecta más a los resultados obtenidos cuanto mayor es la dimensión de las matrices utilizadas hace que disminuya la calidad de las reconstrucciones espectrales [49], [50]. Si el conjunto de espectros que queremos recuperar no queda bien descrito por un número n de vectores base inferior al de sensores k, los anteriores algoritmos limitan la máxima exactitud alcanzable en las reconstrucciones espectrales a la calidad con que $n = k$ vectores describen a dichos espectros.

Shi et al. [36] presentaron un método muy original que trataba de resolver este problema y permitía utilizar más vectores que sensores en las reconstrucciones espectrales sin disminuir por ello la calidad de las mismas.

Si trabajamos con un modelo lineal de representación de espectros, en el que utilizamos n vectores y k sensores, siendo $n > k$, existirá un subespecio de espectros[5] R que registrarán las mismas k respuestas de los sensores[6] q, siendo uno de los espectros de dicho conjunto el que se ha captado con el sistema multiespectral y que, por tanto, debemos tratar de recuperar espectralmente con la mayor exactitud posible. Llamaremos S_R a ese conjunto de espectros compatibles con unas respuestas dadas de los sensores y con el modelo lineal de representación en la base V de n vectores. El método de Shi–Healey tratará de elegir como solución aquel espectro S_R que sea óptimo según algún criterio[7].

Para asociar un único vector[8] R_R perteneciente a S_R a unas respuestas dadas de los sensores q, elegiremos aquel vector que minimice el error cuadrático medio calculado a lo largo de todos los espectros de entrenamiento $[R_{ts}]$ (que es una matriz $N \times m$). En otras palabras, R_R será el vector de S_R que más se parezca espectralmente a alguno de los

[5] Generados al variar los n parámetros σ con que son representados dichos espectros en la base.

[6] De nuevo, metamerismo de la cámara.

[7] Se supone que la solución aportada según este criterio debe mejorar la reconstrucción espectral proporcionada por la pseudoinversión, que es lo que hacían los dos métodos anteriores.

[8] El subíndice $_R$ indica que es la reflectancia reconstruida.

m espectros de entrenamiento, es por ello que la cantidad y calidad[9] de dichos espectros es crucial en este método.

Dado que el sistema multiespectral tiene k sensores, para una dimensionalidad n del modelo lineal, separaremos las contribuciones de los últimos k vectores (que denotaremos con el subíndice 2) de las de los primeros $n-k$ vectores restantes (subíndice 1) en la ecuación (1.7):

$$[q] = [E_Q]^T \left([V_1]\sigma_1 + [V_2]\sigma_2\right) \tag{1.14}$$

Donde $[V_1]$ contiene los vectores base $1, ..., n-k$ y $[V_2]$ contiene los vectores base $n-k+1, ..., n$. Los vectores columna σ_1 y σ_2 contienen los correspondientes coeficientes para la estimación lineal. A partir de la ecuación (1.14) podemos despejar σ_2 en función de σ_1:

$$\sigma_2 = \left([E_Q]^T[V_2]\right)^{-1}\left([q] - [E_Q]^T[V_1]\sigma_1\right) \tag{1.15}$$

Y si sustituimos la última relación en la ecuación (1.4) obtenemos:

$$[R] = [V_1]\sigma_1 + [V_2]\left([E_Q]^T[V_2]\right)^{-1}\left([q] - [E_Q]^T[V_1]\sigma_1\right) \tag{1.16}$$

A partir de la ecuación (1.16), podemos construir una matriz de dimensión $N \times m$ (que llamaremos $[R]^*$) de vectores columna de espectros pertenecientes a S_E para más tarde tratar de escoger aquel vector o espectro óptimo según el criterio anteriormente explicado. Para ello, buscamos una solución por pseudoinversión para calcular σ_1 a partir del conocimiento de los m espectros de entrenamiento:

$$\begin{aligned}[\sigma_1]^* = \Big([V_1] - [V_2]\left([E_Q]^T[V_2]\right)^{-1}[E_Q]^T[V_1]\Big)^+ \cdot \\ \cdot \left([R_{ts}] - [V_2]\left([E_Q]^T[V_2]\right)^{-1}[q]^*\right)\end{aligned} \tag{1.17}$$

[9] Parecido con los espectros que se pretenden reconstruir o estimar.

Donde $[R_{ts}]$ es una matriz $N \times m$ con los m espectros de entrenamientos en sus columnas, $[q]^*$ es una matriz $k \times m$ con las respuestas de los sensores al espectro que queremos estimar repetidas en sus m columnas, y $[\sigma_1]^*$ es una matriz $(n-k) \times m$ que será la que nos genere los m espectros de $[S_E]$ a partir de los cuales buscaremos la estimación espectral $[R_R]$ deseada.

$$[R]^* = [V_1][\sigma_1]^* + [V_2]\left([E_Q]^T[V_2]\right)^{-1}\left([q]^* - [E_Q]^T[V_1][\sigma_1]^*\right) \tag{1.18}$$

De esta forma hemos generado en la matriz $[R]^*$ un conjunto de m espectros, de todos los posibles pertenecientes a S_E, uno de los cuales será la estimación espectral deseada. Como cada columna de $[R]^*$ está relacionada con una columna de $[R_{ts}]$, la estimación espectral elegida será aquella columna de $[R]^*$ que cumpla la condición de mínima distancia euclídea dada por $\|[R]_i^* - R_{tsi}\|$ donde el índice i varía a lo largo de las m columnas de ambas matrices, es decir:

$$[R_R] = [R]_i^* \tag{1.19}$$

Para aquel i que cumpla la anterior condición de mínimo.

La principal desventaja de este método reside en el hecho de que por cada vector de respuestas q debemos calcular m espectros y buscar el mínimo según (1.19). Si m es grande, el método es extremadamente lento [51]. También es imprescindible conocer la sensibilidad Q de los canales de la cámara para poder utilizar este algoritmo. Debemos notar que el algoritmo de Shi–Healey coincide con el de Maloney–Wandell en el caso en que $n = k$, pues la matriz $[V_1]$ sería nula y la ecuación (1.18) coincidiría con la ecuación (1.8) con una matriz $[\Lambda_\epsilon]$ que sería cuadrada $k \times k$.

1.3.4. Algoritmo de Wiener

El algoritmo de estimación espectral de Wiener es de los más ampliamente utilizados por numerosos autores [52], [53]. Al igual que en

el método de Maloney–Wandell, necesitamos conocer la sensibilidad espectral de los sensores de la cámara y un conjunto de espectros de entrenamiento, pero no se utiliza base lineal de representación de dichos espectros. Además, es necesario estimar de forma correcta el ruido que afecta a los sensores de la cámara.

Partiendo de la siguiente ecuación:

$$[q] = [E_Q]^T[R] + [\eta] \tag{1.20}$$

Trataremos de obtener las curvas espectrales R a partir de las respuestas de la cámara q siguiendo un criterio de mínimo error cuadrático medio. Esto se consigue aplicando un operador $[W]$ a las respuestas de los sensores $[q]$ para obtener las estimaciones espectrales $R_R = [W]q$ e imponiendo que:

$$\overline{\|R - R_R\|^2} = \overline{\|R - [W]q\|^2} \tag{1.21}$$

sea mínimo[10]. Se puede demostrar que la forma del operador $[W]$, expresado como una matriz $N \times k$, que cumple la condición impuesta por (1.21) es la siguiente:

$$[W] = [R_{ts}][R_{ts}]^T[Q]\left([Q]^T[R_{ts}][R_{ts}][Q] + [\eta_{ts}][\eta_{ts}]^T\right)^{-1} \tag{1.22}$$

Donde $[E_{ts}]$ es la misma matriz $N \times m$ de la ecuación (1.17) y $[\eta_{ts}]$ es una matriz $k \times m$ con el ruido de cada uno de los k sensores al registrar cada uno de los m espectros de entrenamiento. A la vista de (1.22), queda claro que el producto $[\eta_{ts}][\eta_{ts}]^T$ representa la matriz de autocorrelación del ruido, y si aceptamos la suposición de que el ruido registrado para sensores diferentes en imágenes diferentes no está correlacionado, se puede sustituir la siguiente ecuación en (1.22):

$$[\eta_{ts}][\eta_{ts}]^T = [\eta]^2[I]_k \tag{1.23}$$

[10] $\|\cdot\|^2$ es la norma euclídea. $\bar{\cdot}$ indica promedio.

Donde $[\eta]^2$ es la varianza del ruido de la cámara[11] y la matriz $[I]_k$ es la identidad de dimensiones $k \times k$. Shimano demostró que si el valor de $[\eta]^2$ que se introduce en la ecuación (1.23), y por tanto en la ecuación (1.22), es justamente el medido para la cámara en cuestión, los resultados de las reconstrucciones espectrales obtenidas con este método son óptimos. Esto significa que, para aplicar este método correctamente, se debe haber medido con exactitud el ruido que afecta a la cámara que forma el sistema multiespectral. El propio Shimano propuso un método experimental sencillo para estimar este valor de $[\eta]^2$ óptimo que se debe introducir en (1.22) a partir del análisis de las reconstrucciones espectrales del conjunto de espectros de entrenamiento.

1.3.5. Algoritmo de regresión lineal o de pseudoinversa directa

El método de estimación de regresión lineal, dada su sencillez matemática y su robustez frente al ruido, ha sido utilizado por multitud de autores para reconstruir espectros de reflectancias de objetos y SPD de iluminantes. Este algoritmo también es llamado por algunos autores estimación de Wiener [54], [55], [56], ya que la semejanza matemática entre ambos es considerable. Sin embargo, la información necesaria para aplicar cada uno de ellos es claramente distinta, así como los resultados que se obtienen al aplicar cada uno de ellos.

El algoritmo de regresión lineal es formalmente similar al de Imai–Berns, pero directamente relaciona las respuestas de los sensores $[q_{ts}]$ a los espectros de entrenamiento $[R_{ts}]$ para construir uan matriz de la forma:

$$[D] = [R_{ts}][q_{ts}]^{+} \tag{1.24}$$

La semejanza con el método de Wiener queda de manifiesto si en la ecuación (1.24) sustituimos la (1.20) y desarrollamos la operación de pseudoinversión adecuadamente, ya que en el caso en que $[\eta] = 0$ ambos

[11] La suma al cuadrado de todos los valores *RMS* de los diferentes tipos de ruido existentes.

métodos coincidirían y las ecuaciones (1.24) y (1.22) serían idénticas.

Una vez tengamos la matriz $[D]$ obtenida a partir de los espectros de entrenamiento y sus respuestas registradas en la cámara, podemos estimar otros espectros a partir de las imágenes que nos da el sistema multiespectral:

$$[R_R] = [D][q] \tag{1.25}$$

En este método no se necesita conocer la sensibilidad espectral de los sensores, como en el de Imai–Berns, y ya se incluye el efecto del ruido presente en el sistema en el cálculo de la matriz $[D]$. Pero a diferencia de aquel, este algoritmo no necesita calcular ninguna base lineal de vectores representativos de los espectros que queremos estimar. Debemos notar que, si en el algoritmo de Imai–Berns utilizamos todos los vectores de la base $[V]$ disponibles, de manera que la representación de un espectro de entrenamiento en dicha base es, entonces, exacta:

$$[V]\sigma_{ts} = [R_{ts}] \tag{1.26}$$

Los métodos de regresión lineal y de Imai–Berns son idénticos, como mostramos en la ecuación (1.27), que resulta de manipular las ecuaciones (1.11), (1.12), (1.24), (1.25) y (1.26).

$$[R_R] = [V][A][q] = [V][\sigma_{ts}][q_{ts}]^{+}[q] = [R_{ts}][q_{ts}]^{+}[q] = [D][q] \tag{1.27}$$

El principal inconveniente de este método es que es muy sensible al conjunto de entrenamiento empleado, tanto al número de espectros utilizados como a la forma de los mismos. Diversos autores han propuesto métodos de selección de estos conjuntos de entrenamiento.

1.3.6. Algoritmos de regresión no lineal

Estos algoritmos son una variante del método visto en el apartado anterior siendo lo más normal el tratar de combinar, en forma de

polinomios de mayor o menor orden, las respuestas individuales de cada sensor para tratar mayor grado de información de las imágenes registradas y así aumentar la exactitud de las reconstrucciones obtenidas. Un ejemplo podría ser el siguiente, para el polinomio $1 + X + X^2$ y tres sensores (con sus posibles términos cruzados) [57], [58], [59]:

$$[D_{NL}] = R_{ts} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_1 q_2 \\ q_2 q_3 \\ q_1 q_3 \\ q_1^2 \\ q_2^2 \\ q_3^2 \end{bmatrix} \tag{1.28}$$

En cualquier caso, los resultados no son excesivamente superiores a los conseguidos con el método de regresión lineal, sobre todo en el caso de que se estudien curvas espectrales suaves o de baja dimensionalidad[12]. Dichos resultados varían mucho según la aplicación concreta que se esté estudiando y, sobre todo, con el ruido presente en el sistema ya que, como se ha dicho anteriormente, al aumentar la dimensionalidad del problema (y añadir términos polinómicos lo hace) el ruido afecta en mayor medida a la estimación espectral proporcionada por el algoritmo.

1.4. Ruido en los sistemas de captura

Las fuentes de ruido inherentes a los sistemas de captura alteran los valores digitales de medida del color en cada píxel, distorsionan de forma desconocida la imagen real capturada y degradan su precisión radiométrica, su calidad y su resolución.

A continuación se realiza una breve reseña de los tipos de ruidos más

[12]Que quedan bien descritas con pocos vectores PCA.

relevantes y de sus características, así como los métodos para corregir sus efectos.

1.4.1. Clasificación de los ruidos

Los distintos tipos de ruidos pueden ser clasificados en dos grandes grupos: *ruidos temporales* y *ruidos espaciales*. El efecto producido por los ruidos temporales puede ser corregido promediando fotogramas pero no ocurre así con el efecto producido por los ruidos espaciales [60], [61]. Este debe ser corregido aplicando técnicas de corrección de ganancia u offset.

Dentro del ruido temporal encontramos:

- Ruido de disparo
- Ruido de disparo de corriente oscura
- Ruido de lectura
- Ruido de reset

Dentro del ruido espacial encontramos:

- Heterogeneidad de corriente oscura
- Heterogeneidad en la respuesta (PRNU, Photo Response Non-Uniformity), la cual da lugar al ruido de patron fijo (FPN, Fixed Pattern Noise).

A continuación se realizará una breve descripción de los diferentes tipos de ruidos:

Ruido de disparo

El ruido de disparo está asociado a la llegada aleatoria de fotones al sensor. Este ruido representa el límite natural en el rendimiento de los sensores.

El tiempo que transcurre entre la llegada de los fotones sigue una distribución de Poisson, por lo tanto el ruido de disparo es igual a la raíz cuadrada de la señal expresada en electrones.

Ruido de lectura

El ruido de lectura es un ruido electrónico independiente de la exposición que se añade a la señal final durante la transformación de carga en tensión, previa a la conversión analógica–digital.

El ruido de lectura puede ser aislado y eliminado sustrayendo la imagen 'bias'. Una imagen 'bias' es una imagen capturada sin exposición[13] que permite aislar el ruido de lectura porque lleva a la luz y a la temperatura generada por los circuitos a sus niveles mínimos.

Ruido de cuantización

El convertidor analógico–digital (ADC) produce valores discretos de salida digital, por ello existe cierto rango de tensiones diferentes que producen la misma salida. Este es un ruido aleatorio independiente de la señal con media igual a cero.

Ruido de reset

Aparece cuando se resetea el sensor.

Ruido de corriente oscura

La corriente oscura es el resultado de imperfecciones o impurezas en el silicio o en la interfaz de dióxido de silicio–silicio. Físicamente, la corriente oscura se debe a la generación aleatoria de huecos y electrones –dentro de la región de agotamiento– que son barridos por el elevado campo eléctrico.

La generación de corriente oscura es un proceso térmico en el que los electrones utilizan energía térmica para alcanzar un estado intermedio, desde donde son emitidos a la banda de conducción y añadidos a la señal medida por el píxel. Por esta razón la forma más eficaz de reducir la corriente oscura es enfriar el sensor, evitando de esta manera que los electrones alcancen un estado intermedio. La corriente oscura es

[13] Tiempo de exposición igual a cero.

multiplicativa: su nivel es proporcional al tiempo de exposición y genera dos tipos de ruido: *Heterogeneidad de corriente oscura (FPN)*[14] y *Ruido de disparo de corriente oscura*

La heterogeneidad de corriente oscura o el ruido de patrón fijo se debe a que cada píxel genera una cantidad de corriente oscura ligeramente distinta al resto de píxeles debido a sutiles diferencias en el tamaño del detector, en la densidad del dopaje y en los elementos extraños que puedan quedar atrapados durante el proceso de fabricación. Este ruido puede ser eliminado sustrayendo un fotograma de referencia oscura tomado a la misma temperatura y con el mismo tiempo de integración.

El ruido de disparo de corriente oscura es un tipo de ruido de disparo, por lo tanto será igual a la raíz cuadrada de la señal oscura expresada en electrones. El ruido oscuro en la imagen resultante de sustraer el fotograma oscuro a la imagen en crudo es mayor que esta en un factor de $\sqrt{2}$.

Heterogeneidad en la respuesta (PRNU)

La heterogeneidad en la respuesta es consecuencia de las diferencias en la sensibilidad a la luz que presentan los diferentes píxeles debidas a variaciones en el proceso de fabricación. El resultado a nivel de píxeles es un patrón de "tablero de ajedrez" en una imagen plana. Habitualmente esta variación es del orden de un 2 % de la señal media y mantiene una relación lineal con ella.

La heterogeneidad en la respuesta depende de la señal y puede ser corregida capturando una imagen uniforme que se utiliza para calibrar las diferencias entre los píxeles. Aunque este proceso elimina el PRNU, la sustracción de la imagen introduce un factor de incremento de $\sqrt{2}$ en el ruido de disparo.

El ruido se mide habitualmente en unidades equivalentes de electrón. La magnitud del ruido se especifica en valor RMS[15] sobre el proceso

[14] Fixed Pattern Noise.
[15] Root Mean Square.

aleatorio que causó el ruido. Puesto que no existe correlación entre las distintas fuentes de ruido, el ruido total viene dado como suma de la potencia cuadrática de los distintos tipos de ruido.

1.4.2. Corrección del ruido

La obtención de una imagen promediada elimina todas las fuentes de ruido[16] excepto FPN y PNRU [62], [63], [64]. Estos dos tipos de ruidos dan lugar a la heterogeneidad espacial de respuesta, la cual debe corregirse si lo que se pretende es utilizar la cámara como un instrumento de medida con alta resolución espacial.

Existen dos tipos de técnicas empleadas para corregir la heterogeneidad espacial:

- Corrección de campo uniforme (calibración): Consiste en capturar –además de la imagen que se desea corregir– dos imágenes adicionales: una oscura y otra uniforme. La imagen corregida será combinación lineal de la imagen sin corregir, la imagen oscura y la imagen uniforme.

- Técnicas de escena: Consisten en aplicar un algoritmo a la imagen original para obtener una mejora en la calidad de imagen a expensas de la precisión radiométrica. Hay una gran variedad de técnicas de este tipo, dependiendo del algoritmo aplicado: algoritmos estadísicos (Hayat et al. [65]), algoritmos de movimiento (Hardie et al. [66]), algoritmos algebraicos (Ratliff et al. [67]) y método de la forma de la covarianza inversa (Torres et al. [68]).

Ratliff et al. [69] ha desarrollado una técnica que combina la calibración con un algoritmo algebraico dando lugar a un tipo de corrección de escena con precisión radiométrica.

También se han empleado técnicas de filtrado vectorial para eliminar el ruido en las imágenes de color. Uno de los campos de aplicación de este tipo de técnicas es la astronomía, ya que requiere imágenes de alta calidad

[16] Debido al origen de dichas fuentes y a sus características fundamentales.

y no ofrece la posibilidad de proceder por calibración (Oppenheim et al.[70]).

La corrección de la heterogeneidad espacial por calibración permite utilizar una cámara digital como instrumento para la medida del color con una alta resolución espacial y con precisión radiométrica. Básicamente existen dos variantes de las técnicas de corrección de campo uniforme y estas dos variantes requieren dos imágenes: una imagen oscura con el objetivo cubierto capturada a la misma temperatura y con el mismo tiempo de exposición que la imagen a corregir y una imagen de campo uniforme.

Primera variante de la corrección de campo uniforme

En la primera variante de corrección de heterogeneidad espacial propuesta por Tyson et al. [71], la imagen de campo uniforme corresponde a la imagen de una superficie gris o a una pantalla iluminada colocada exactamente en el mismo lugar donde se van a poner las imágenes a corregir, con la misma iluminación y con el mismo tiempo de exposición. Asumiendo que la superficie gris (o la pantalla iluminada) es completamente uniforme, esta será utilizada para compensar numéricamente la heterogeneidad espacial (Thomson et al. [72]). Esto se consigue capturando una imagen de campo uniforme a la que se resta la imagen oscura con el mismo tiempo de exposición.

En el caso en que se capturen varias imágenes usando filtros, se deberá obtener una imagen de campo uniforme para cada filtro. En este caso, la heterogeneidad espacial debida tanto a la iluminación como a la respuesta del dispositivo contribuyen a la heterogeneidad espacial de la imagen, por lo que cualquier cambio en las condiciones de iluminación requiere capturar una nueva imagen de campo uniforme. El proceso de calibración queda descrito por la ecuación (1.29):

$$I_c = \frac{I - I_{dark}}{I_{unif}} \tag{1.29}$$

Donde I_c representa la imagen corregida, I representa la imagen

original sin corregir, I_{dark} representa la imagen oscura e I_{unif} representa la imagen de campo uniforme (corregida por la imagen oscura).

En el caso de applicaciones con bajo nivel de iluminación, también se puede aplicar una corrección de imagen 'bias', ya que el ruido de lectura podría no ser despreciable frente a la señal medida, al contrario que en situaciones de iluminación media o alta, en las cuales se considera que la imagen 'bias' está incluida en la imagen oscura debido a su pequeña importancia en relación a la imagen a corregir. Cuando se aplica la corrección de imagen 'bias', la imagen de campo uniforme usada en la calibración se obtiene restando la imagen 'bias' y la correspondiente imagen oscura a la imagen de campo uniforme no corregida. En este caso, el proceso de calibración queda descrito por la ecuación (1.30):

$$I_c = \frac{I - I_{bias} - I_{dark}}{I_{unif}} \tag{1.30}$$

Donde I_c representa la imagen corregida, I representa la imagen original sin corregir, I_{bias} representa la imagen 'bias', I_{dark} representa la imagen oscura e I_{unif} representa la imagen de campo uniforme (corregida por la imagen 'bias' y por la imagen oscura).

Segunda variante de la corrección de campo uniforme

En la segunda variante de corrección de heterogeneidad espacial, la imagen de campo uniforme corresponde a la imagen de un campo de radiancia uniforme que en adelante será llamada imagen de corrección base. La imagen de corrección base que se emplea habitualmente es la imagen brillante que se define como la imagen con el valor digital medio más elevado posible que no satura ningún píxel. La imagen oscura y la imagen de corrección base se combinan con la imagen a corregir por medio de dos algoritmos lineales.

En el primer algoritmo, las imágenes son directamente combinadas de acuerdo a la ecuación (1.31):

$$DL_c(i,j) = k\frac{DL(i,j) - DL_0(i,j)}{DL_B(i,j) - DL_0(i,j)} \tag{1.31}$$

Donde $DL_c(i,j)$, $DL(i,j)$, $DL_0(i,j)$ y $DL_B(i,j)$ son los valores digitales en el píxel (i,j) de la imagen corregida, la imagen original sin corregir, la imagen oscura y la imagen de corrección base respectivamente, y k es la constante de calibración que se estima por el nivel digital medio de la imagen resultante de la diferencia entre la imagen de corrección base y la imagen oscura.

El segundo algoritmo[17] está basado en el cálculo de las matrices de ganancia y offset dadas por las ecuaciones (1.32):

$$\begin{aligned} DL_c(i,j) &= O(i,j) + G(i,j)DL(i,j) \\ G(i,j) &= \frac{DL_B - DL_0}{DL_B(i,j) - DL_0(i,j)} \\ O(i,j) &= DL_0 - G(i,j)DL_0(i,j) \end{aligned} \tag{1.32}$$

Donde $O(i,j)$ con $i = 1,...,m$ y $j = 1,...,n$ representa el elemento (i,j) de la matriz de corrección de offset O, $G(i,j)$ representa el elemento (i,j) de la matriz de corrección de ganancia G y DL_0 y DL_B son los niveles digitales de referencia de la imagen oscura y de la imagen de corrección base respectivamente. El valor digital medio para todos los píxeles de la imagen es utilizado habitualmente como valor digital de referencia.

El tamaño de las matrices $(m \times n)$ depende del número de píxeles disponible en el sensor[18].

Las diferencias entre los elementos de la matriz de corrección de offset O y la matriz de ganancia G se deben a pequeñas fluctuaciones en las respuestas de cada pixel. Estas fluctuaciones causan el ruido de patrón fijo[19] (FPN) que se pretende corregir con el algoritmo lineal mencionado.

[17] Empleado por de Lasarte [73].

[18] m corresponde al número de píxeles presentes en una fila del sensor y n corresponde al número de píxeles en cualquier columna.

[19] Este ruido es inherente a la respuesta de la cámara.

Como se mencionó anteriormente, el valor de referencia digital y la imagen de corrección base habitualmente utilizados son el valor digital medio y la imagen brillante respectivamente. Aunque usando estos elementos se obtienen resultados aceptables, la corrección de heterogeneidad podría ser significativamente mejorada aplicando el algoritmo lineal con otros elementos. La influencia de la imagen oscura –que puede tomar un valor nulo o no nulo–, la imagen de corrección base –que puede ser diferente de la imagen brillante– y el valor de referencia digital –que puede ser distinto a la media utilizada habitualmente– se ha estudiado para optimizar la corrección de heterogeneidad espacial.

1.4.3. Tratamiento del ruido en el presente trabajo

En el presente trabajo no se pretende construir un dispositivo con alta resolución espacial, por esta razón el ruido será tratado realizando un promediado de los colores de los píxeles contenidos en cuadros de 36x36 píxeles para cada parche de color tal y como se describe al final de 2.1.3. Otra razón que apoya esta forma de tratar el ruido es que se buscan resultados relativos entre los métodos de reconstrucción, no resultados absolutos.

Capítulo 2

Método

En el presente capítulo se describen procedimientos generales comunes para todos los métodos de reconstrucción que se emplean y posteriormente se detalla método a método cuales son los pasos a seguir para realizar la reconstrucción. Dentro de los procedimientos generales se explica como obtener el software, como medir espectralmente los parches de color para el entrenamiento del sistema, como realizar las tomas fotográficas, como convertir los datos entre los distintos espacios de color, como medir la bondad o el error cometido en la reconstrucción y como convertir la información entre los distintos tipos de archivo que usan las aplicaciones. Los métodos empleados para realizar la reconstrucción que se describen en este capítulo son: método de la pseudoinversa directa, método de la descomposición en valores singulares, método del análisis de componentes independientes, método de los colores cercanos y método cuadrático de los colores cercanos.

2.1. Procedimiento general

El procedimiento general para realizar una reconstrucción espectral por cualquiera de los métodos –ver Klein [74]– que se describen en el presente capítulo consiste en obtener los valores espectrales del conjunto de parches de entrenamiento y del conjunto de parches de test[1] según se indica en la sección 2.1.2, así como los valores tricrómaticos RGB de ambos conjuntos de parches según se indica en la sección 2.1.3.

En el presente trabajo se han utilizado los parches neutro A2, azul B5, verde D9, rojo L3 y amarillo M7 de la carta de color ColorChecherSG[2] como parches de test[3] y el resto de parches de la misma carta como parches de entrenamiento[4] [75].

Posteriormente hay que convertir la lectura espectral y tricromática de los parches que se encuentran en un fichero .txt a variables matriz de MATLAB como se indica en la sección 2.1.10.

Una vez disponemos de la información tricromática y espectral de los parches en variables matriz de MATLAB, tenemos que transformar los valores tricromáticos RGB al espacio XYZ, siguiendo las instrucciones que se detallan en las secciones 2.1.4 y 2.1.5. Para convertir valores sRGB al espacio XYZ se enumeran diferentes formas de hacerlo, pero se recomienda emplear la norma IEC 61966-2-1:1999[5]. También puede requerirse la transformación de valores del espacio XYZ al espacio Lab para la aplicación del método explicado en la sección 2.5.

Llegados a este punto, ya es posible trabajar con los datos recogidos y aplicar los diferentes métodos de reconstrucción aquí expuestos y detallados en las secciones 2.2, 2.3, 2.4, 2.5 y 2.6 para predecir las curvas

[1] En principio no sería necesario obtener la reflectancia espectral de los parches de test puesto que su predicción es el objeto del presente trabajo de investigación, pero sí que es necesaria para tener y referencia y poder medir la bondad de las reconstrucciones.

[2] Más información sobre la misma en http://xritephoto.com/(S(gmv0gg45nfrsqbyhrsmzvxza))/ph_product_overview.aspx?id=938&catid=28&action=overview.

[3] En total, 5 parches de test.

[4] En total, 135 parches de entrenamiento.

[5] Más información en http://www.color.org/srgb.pdf y dentro de la sección 2.1.4.

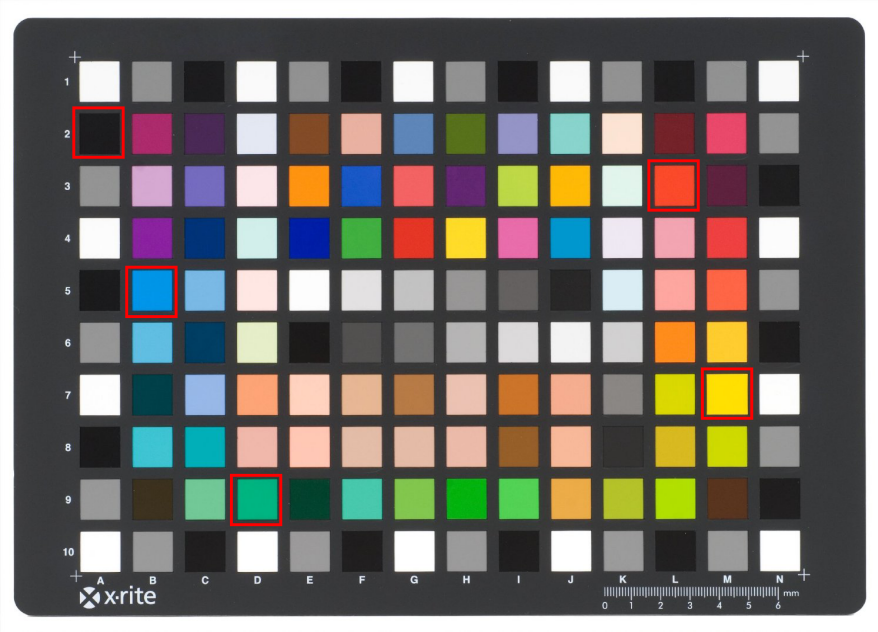

Figura 2.1: Carta ColorCheckerSG sobre la que se han señalado en recuadros rojos los parches de prueba. El resto de parches se utilizan como parches de entrenamiento.

de reflectancia espectral de los parches.

Para finalizar solo resta medir el error cometido en la reconstrucción de los parches de test, y comparar ese error entre los distintos métodos empleados, tal y como se describe en la sección 2.1.7. Para verificar que el resultado empleando dos iluminantes distintos supone una mejoría también se puede comparar con el que se obtendría empleando un solo iluminante.

2.1.1. Obtención del software

El software propietario MATLAB puede adquirirse en la página oficial de Mathworks `http://www.mathworks.es/products/matlab/`.

El software propietario Photoshop puede adquirirse en la página

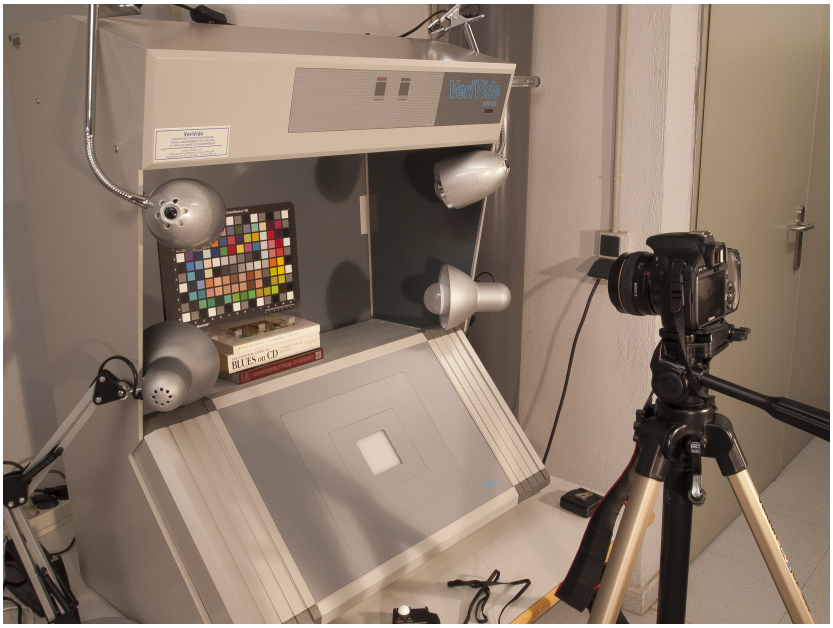

Figura 2.2: Fotografía de perfil del dispositivo experimental empleado en el laboratorio para capturar los valores tricromáticos de los parches de color.

oficial de Adobe http://www.adobe.com/es/products/photoshop.html.

El software freeware de Photivo puede descargarse de su página oficial http://photivo.org/

El software freeware ColorLab puede descargarse gratuitamente de la página de xRite http://www.xrite.com/product_overview.aspx?ID=1071&Action=support&SoftwareID=486. ColorLab está disponible para Windows y para MacOS.

Para poner en práctica el método descrito en la sección 2.4 se necesitará el paquete de MATLAB FastICA que puede descargarse de forma gratuíta en la página web http://research.ics.aalto.fi/ica/fastica/.

La versión de ColorLab para MacOS requiere arquitectura PowerPC (máquinas de Apple entre 1994 y 2006) o en su defecto Rossetta[6]. El

[6]Aplicación de MacOS para que la arquitectura nueva de Intel pueda correr aplicacio-

Figura 2.3: Fotografía frontal del dispositivo experimental empleado en el laboratorio para capturar los valores tricromáticos de los parches de color.

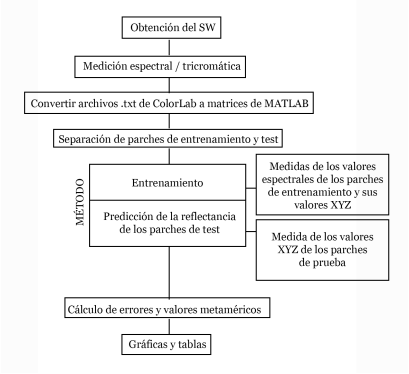

Figura 2.4: Esquema general del procedimiento a seguir para realizar la reconstrucción por cada uno de los diferentes métodos.

nes para PowerPC.

problema es que Rossetta funciona solo en ordenadores de Apple con procesadores Intel y sistemas operativos previos a MacOS 10.7 (Lion)[7]. A partir de Lion ya no puede instalarse Rossetta[8] por lo que no resulta a priori posible disponer de ColorLab en un ordenador Mac con sistema operativo MacOS 10.7 o posterior. Una solución posible es emplear un PC con Windows en lugar de Mac. Otra solución[9] es virtualizar un sistema operativo Windows dentro de MacOS e instalar en ese sistema operativo virtualizado la versión de Windows. Otra solución es emplear una aplicación para ejecutar programas de Windows dentro de Mac como CrossOver o Wineskin.

La solución por la que se ha optado en el presente trabajo es la de usar MacOS 10.9.3 (Mavericks) e instalar la versión de ColorLab para Windows usando Wineskin.

2.1.2. Medición espectral

Se abre la aplicación ColorLab. Se crea una nueva carta de color (File → New). Hay que cerciorarse de que el espectrofotómetro está debidamente conectado (normalmente al puerto USB). En Edit → Instrument configuration... se selecciona el modelo de espectrofotómetro que se va a emplear para realizar las mediciones y se activa la casilla "spectral" (para que las mediciones sean espectrales).

Se crea una (solo una) columna de parches de color con la misma longitud que la longitud de la carta original en Edit → New Patches. Por ejemplo, para la carta ColorChecker SG se pondría en Columns el valor 1 y en Rows el valor 10, generando así un conjunto de 10 parches organizados en 1 columna. El resto de columnas las generará automáticamente el programa conforme vayan realizándose las mediciones.

A continuación Filter→ Measure. Es posible que se tenga que medir en primer lugar el blanco de referencia que se incluye con el espectrofotómetro antes de empezar.

[7] Previos a octubre de 2010.

[8] Apple alegó razones de seguridad.

[9] Quizás sería más apropiada la palabra "remiendo" que "solución" en este caso.

Cuando lo pida la aplicación, se situa el espectrofotómetro sobre el primer parche y se presiona el botón de hardware que indica a la máquina que se va a realizar una medición. Cuando esté hecha la medición, se pasa al segundo parche y así sucesivamente hasta completar la primera columna de parches. Después se continuaría con la segunda columna de parches y con la tercera, etc... hasta terminar de medir toda la carta.

Cuando toda la carta está medida, File → Save as... y se guarda el archivo de carta de color que se ha medido como fichero de extensión .txt.

2.1.3. Medición tricromática

Tomas fotográficas y revelado

Cada fotografía que se realiza debe tener los mismo ajustes [76]. Lo único que debe cambiar es el iluminante. Además, el valor de brillo en cd/m^2 medido sobre la superficie de la carta debe ser el mismo para cada iluminante. Para ello se ajustará la potencia de los iluminantes con ayuda de un luxómetro. El revelado de todas las fotografías debe ser idéntico y las mediciones de la cámara deben quedar todas dentro del histograma[10].

Es aconsejable empezar haciendo la primera foto con el iluminante de mayor temperatura y procurar que el histograma de dicha foto quede alineado a la derecha, para dejar el espacio de la izquierda a las fotografías con iluminantes de menor temperatura. Para evitar que la imagen se salga del histograma conviene emplear una cámara fotográfica que tenga un buen rango dinámico. También hay que tener en cuenta que el rango dinámico de algunas cámaras puede disminuir si se emplean valores de ISO demasiado bajos, aunque por lo general es conveniente usar un ISO bajo para evitar el ruido. Si la cámara tiene buen rango dinámico, se puede emplear el valor de ISO más bajo posible. Si la cámara tiene un rango dinámico estrecho, puede emplearse el segundo valor más bajo posible

[10]El histograma es una representación gráfica de la frecuencia de los valores RGB en cada píxel que puede obtenerse en la pantalla de la cámara en el momento en que se va a realizar el disparo.

de ISO. El perfil de color de la cámara puede ser sRGB o AdobeRGB.

Se revelan las fotografías con ayuda de algún programa de revelado fotográfico. Para el presente trabajo de investigación se han realizado pruebas preliminares con Adobe Camera Raw[11] y con Photivo. Al final se ha optado por usar los resultados obtenidos por Photivo ya que cuenta con mayor cantidad de ajustes para el revelado y trabaja a un nivel más bajo.

La salida del revelado se hace en formato TIFF sin compresión.

Unos posibles ajustes de revelado empleando Adobe Camera Raw serían los que se detallan en la figura 2.1 y unos posibles ajustes de revelado empleando Photivo [12] serían los que se detallan en le figura 2.2.

Convertir la revelada en fichero de ColorLab

Con ayuda de Photoshop se irán recortando rectángulos de 36x36 píxeles dentro de cada parche de la imagen fotografiada y se irán colocando en orden dando lugar a una nueva versión de la carta con dimensiones de 504x360 píxeles. Para promediar los valores de los distintos píxeles de cada parche y de esta manera corregir el ruido temporal de la imagen se abre esta nueva versión de la carta realizada con Photoshop en ColorLab y se hace click en Filter → Layout and Format → Spot Colors. Se introducen las dimensiones de la carta (14 parches de ancho por 10 parches de alto) y se presiona en aceptar. Posteriormente se guarda la carta como fichero .txt de ColorLab.

La razón por la que los recortes de cada parche se realiza con dimensiones de 36x36 es debido a los límites de resolución de la cámara empleada y porque de esta manera se promedian 1296 mediciones de color distintas que parecen suficientes.

[11] Plugin de Photoshop.

[12] Los ajustes no especificados en la figura 2.2 se pondrán a valor de reset.

Atributo	Valor
Temperatura de color	5000ºK
Tinta	+23
Exposición	-1.00
Recuperación	0
Luz de relleno	0
Negros	0
Brillo	+50
Contraste	+25
Resonancia (vibrance)	0
Saturación	0
Curva tonal	Lineal 1 a 1
Cantidad de nitidez	25
Radio de nitidez	1
Detalle de nitidez	25
Máscara de nitidez	0
Reducción de ruido de luminancia	0
Reducción de ruido de color	25
Detalle de color	50
Tono (H), Saturación (S) y Luminancia (L)	Todos a 0
Tono de luces	0
Saturación de luces	0
Balance	0
Tono de sombras	0
Saturación de sombras	0
Grano	0
Viñeteado	0
Profundidad de color	16 bits

Tabla 2.1: Detalle de los ajustes empleados para el revelado de las imágenes empleando el plugin de Photoshop, Adobe Camera RAW.

2.1.4. Convertir valores sRGB al espacio XYZ

Los algoritmos de reconstrucción de curvas espectrales requieren –entre otros– los valores XYZ de la carta de color para el entrenamiento puesto que estos valores son independientes de cualquier dispositivo,

Atributo	Valor
Camera profile	flat profile
Rendering intent	absolute colorimetric
White balance	5737ºK
Highlights	$R = 0$, $G = 0$, $B = 0$
Color intensity	all to 0
Brightness	all to 0
Exposure	1.8
RGB curve	linear
Rendering intent	absolute colorimetric
After gamma curve	linear
Wiener sharpen	reset settings

Tabla 2.2: Detalle de los ajustes empleados para el revelado de las imágenes empleando Photivo.

entonces se necesita transformar los valores del espacio sRGB a valores XYZ (Wandell et al. [77]).

Existen muchas maneras de realizar esta transformación, pero se comprueba que cada una de ellas arroja resultados diferentes.

De sRGB a XYZ con MATLAB

La aplicación MATLAB incorpora una librería para el procesamiento de imágenes http://www.mathworks.es/products/image/. El conjunto de sentencias que nos permite realizar la transformación de sRGB a XYZ es:

```
Irgb = imread('nombre-del-archivo.tif');
C = makecform('srgb2xyz');
Ixyz = applycform(Irgb,C);
imwrite(Ixyz, 'nuevo-nombre-de-archivo.tif')
```

No obstante se comprueba que la transformación siguiendo este método no es muy satisfactoria. En la figura 2.5 que se muestra a

continuación[13] se observa a simple vista y con total claridad que el color no permanece constante durante la transformación.

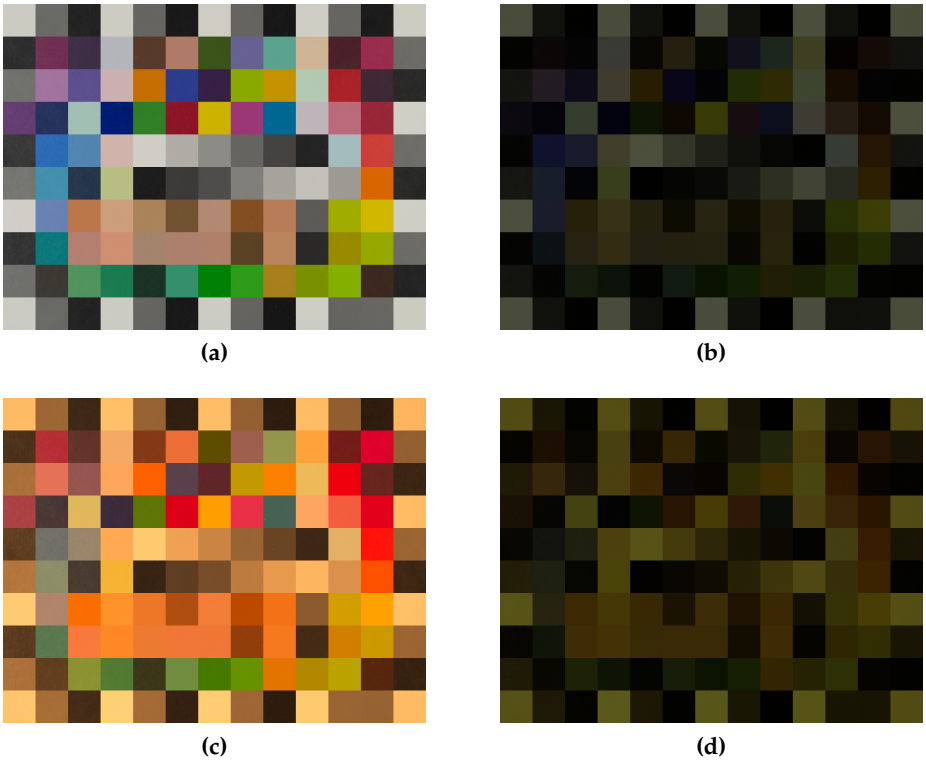

Figura 2.5: Image Processing Toolbox de MATLAB hace una "extraña" transformación del espacio sRGB al espacio XYZ. La figura (a) muestra la carta de color ColorChecker SG con iluminación LED en el espacio sRGB antes de transformarla al espacio XYZ. La figura (b) muestra la misma carta de color después de haber sido transformada al espacio XYZ con Image Processing Toolbox de MATLAB. La figura (c) muestra la carta de color ColorChecker SG con iluminación incandescente en el espacio sRGB antes de transformarla al espacio XYZ. La figura (d) muestra la misma carta de color después de haber sido transformada al espacio XYZ con Image Processing Toolbox de MATLAB. Se comprueba a simple vista que la transformación es un poco "extraña".

[13]Sirvan como ejemplo a pesar de que el ruido de las mismas no ha sido debidamente corregido.

De sRGB a XYZ con ColorLab

ColorLab dispone de una opción para hacer el cambio que se encuentra en el menú Filter → Mode → XYZ. Realizando la transformación de esta manera se consiguen resultados mucho mejores que con MATLAB. El problema es que desconocemos cual es el algoritmo que sigue el programa para realizar el cambio.

De sRGB a XYZ usando un perfil de color ICC

El problema de utilizar un perfil de color ICC para la transformación del espacio sRGB al XYZ es que las aplicaciones que usan perfiles de color necesitan realizar ajustes e interpolaciones sobre las matrices del perfil, de tal modo que usando el mismo perfil en distintas aplicaciones pueden obtenerse distintos resultados.

De sRGB a XYZ con la norma IEC 61966-2-1:1999

Este es, de todos los métodos expuestos para pasar de sRGB a XYZ el más apropiado porque siguiendo un sencillo algoritmo se logra un cambio satisfactorio manteniendo los colores constantes. Además es conocido y está normalizado[14].

$$\begin{aligned} a &= 0{,}055 \\ \gamma &= 2{,}4 \end{aligned} \tag{2.1}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}412424 & 0{,}357579 & 0{,}180464 \\ 0{,}212656 & 0{,}715158 & 0{,}072186 \\ 0{,}019332 & 0{,}119193 & 0{,}950444 \end{bmatrix} \begin{bmatrix} g\left(\frac{R}{255}\right) \\ g\left(\frac{G}{255}\right) \\ g\left(\frac{B}{255}\right) \end{bmatrix} \tag{2.2}$$

Donde:

[14] Más detalles en http://www.colour.org/tc8-05/Docs/colorspace/61966-2-1.pdf.

$$g(k) = \left(\frac{k+a}{1+a}\right)^{\gamma} \tag{2.3}$$

para k>0,04045. Si no:

$$g(k) = \frac{k}{12{,}92} \tag{2.4}$$

A continuación se muestra una rutina de MATLAB que implementa esta transformación:

```
function [Y]=srgb2xyz(X)
% rgb2xyz que transforma una matriz X cuyas filas contienen datos
% RGB en una matriz Y cuyas filas contienen datos XYZ segun la
% norma IEC 61966-2-1:1999


a=0.055;
gamma=2.4;
T1=[0.412424 0.357579 0.180464];
T2=[0.212656 0.715158 0.072186];
T3=[0.019332 0.119193 0.950444];
T=[T1; T2; T3];

% Se podria haber implementando esta rutina con menos linas de
% codigo, pero se ha optado por esta forma de implementacion para
% mejorar la legibilidad

kr=X(1, 1)/255;
kg=X(1, 2)/255;
kb=X(1, 3)/255;

if kr > 0.04045
```

```
    gkr=100*(((kr+a)/(1+a))^gamma);
else
    gkr=100*(kr/12.92);
end

if kg > 0.04045
    gkg=100*(((kg+a)/(1+a))^gamma);
else
    gkg=100*(kg/12.92);
end

if kb > 0.04045
    gkb=100*(((kb+a)/(1+a))^gamma);
else
    gkb=100*(kb/12.92);
end

temp=T*[gkr; gkg; gkb];
Y=[temp'];

for i=2:size(X, 1)

    kr=X(i, 1)/255;
    kg=X(i, 2)/255;
    kb=X(i, 3)/255;

    if kr > 0.04045
        gkr=100*(((kr+a)/(1+a))^gamma);
    else
        gkr=100*(kr/12.92);
    end

    if kg > 0.04045
        gkg=100*(((kg+a)/(1+a))^gamma);
```

```
    else
        gkg=100*(kg/12.92);
    end

    if kb > 0.04045
        gkb=100*(((kb+a)/(1+a))^gamma);
    else
        gkb=100*(kb/12.92);
    end

    temp=T*[gkr; gkg; gkb];
    Y=[Y; temp'];

end

end
```

2.1.5. Convertir valores AdobeRGB a XYZ

Puede encontrarse información detallada sobre la transformación del espacio AdobeRGB al espacio XYZ y viceversa en la Página web del International Color Consortium[15] o también en la página web de Adobe[16].

Sean los valores R', G' y B' los obtenidos por la cámara digital escalados entre 0 y 255. Estos se transforman en valores triestímulo RGB entre 0 y 1 mediante:

[15] Más detalles en http://www.color.org/adobergb.pdf.
[16] Más detalles en http://www.adobe.com/digitalimag/pdfs/AdobeRGB1998.pdf.

$$
\begin{aligned}
R &= \left(\frac{R'}{255}\right)^{2,199} \\
G &= \left(\frac{G'}{255}\right)^{2,199} \\
B &= \left(\frac{B'}{255}\right)^{2,199}
\end{aligned}
\tag{2.5}
$$

Posteriormente se transforman a XYZ mediante:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}57667 & 0{,}18556 & 0{,}18823 \\ 0{,}29735 & 0{,}62736 & 0{,}07529 \\ 0{,}02703 & 0{,}07069 & 0{,}99133 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.6}
$$

A continuación se muestra una rutina de MATLAB que permite transformar los datos del espacio AdobeRGB al espacio XYZ:

```
function [Y]=adobergb2xyz(X)
% adobergb2xyz transforma una matriz X cuyas filas contienen
% datos AdobeRGB en una matriz Y cuyas filas contienen datos
% XYZ. Pueden encontrarse más detalles aquí:
% http://www.adobe.com/digitalimag/pdfs/AdobeRGB1998.pdf

exp=2.199;
T1=[0.57667 0.18556 0.18823];
T2=[0.29735 0.62736 0.07529];
T3=[0.02703 0.07069 0.99133];
T=[T1; T2; T3];
```

```
kr=(X(1, 1)/255)^exp;
kg=(X(1, 2)/255)^exp;
kb=(X(1, 3)/255)^exp;

temp=T*[kr; kg; kb];
Y=[temp'];

for i=2:size(X, 1)

    kr=(X(i, 1)/255)^exp;
    kg=(X(i, 2)/255)^exp;
    kb=(X(i, 3)/255)^exp;

    temp=T*[kr; kg; kb];
    Y=[Y; temp'];

end
Y=100.*[Y];
end
```

2.1.6. Convertir valores XYZ a Lab

La transformación del espacio XYZ al espacio Lab exige la consideración de las coordenadas normalizadas XYZ de un punto blanco de referencia. Habitualmente se consideran las coordenadas XYZ del D50 como blanco de referencia.

$$\begin{aligned} X_n &= 96{,}4212 \\ Y_n &= 100{,}0 \\ Z_n &= 82{,}5188 \end{aligned} \tag{2.7}$$

$$
\begin{aligned}
L^* &= 116 f\left(\frac{Y}{Y_n}\right) - 16 \\
a^* &= 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \\
b* &= 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)
\end{aligned}
\tag{2.8}
$$

Donde:

$$
f(t) = \begin{cases} t^{\frac{1}{3}} & \text{si } t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{en otro caso} \end{cases}
\tag{2.9}
$$

A continuación se muestra una rutina de MATLAB que permite transformar los datos del espacio XYZ al espacio Lab:

```
function [Y]=xyz2Lab(X)
% xyz2Lab transforma una matriz X cuyas filas contienen
% datos XYZ en una matriz Y cuyas filas contienen datos Lab

XN = 96.4212;
YN = 100.0;
ZN = 82.5188;

tx=X(1, 1)/XN;
ty=X(1, 2)/YN;
tz=X(1, 3)/ZN;

L=116*preventInfSlope(ty)-16;
```

```
a=500*(preventInfSlope(tx)-preventInfSlope(ty));
b=200*(preventInfSlope(ty)-preventInfSlope(tz));

Y=[L a b];

for i=2:size(X, 1)

    tx=X(i, 1)/XN;
    ty=X(i, 2)/YN;
    tz=X(i, 3)/ZN;

    L=116*preventInfSlope(ty)-16;
    a=500*(preventInfSlope(tx)-preventInfSlope(ty));
    b=200*(preventInfSlope(ty)-preventInfSlope(tz));

    Y=[Y; [L a b]];

end
    function kx=preventInfSlope(zx)

        tlimit=(6/29.0)^3;

        if zx > tlimit
            kx=zx^(1/3.0);
        else
            kx=(1/3.0)*((29/6.0)^2)*zx+(4/29.0);
        end
    end
end
```

2.1.7. Cálculo de errores

La medida más evidente del error cometido en la reconstrucción de los parches es la diferencia de color entre el parche original y el reconstruido. No obstante, existen varias formas de medir la diferencia entre los colores. En el presente trabajo de investigación se emplea error ΔE [78] como medida perceptiva del color y el error cuadrático medio RMS como medida de la diferencia entre los espectros.

Error ΔE

El error ΔE es una medida perceptiva de la diferencia de color que originalmente se derivó del espacio de color CIELab como la distancia euclídea entre dos puntos en ese espacio.

Sean L_1, a_1, b_1 y L_2, a_2, b_2 las coordenadas Lab de dos colores. Su diferencia $\Delta E76$[17] se calcula como:

$$\Delta E = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2} \tag{2.10}$$

Esta definición de ΔE no es la más precisa de todas, pero sí es la más sencilla de obtener y proporciona una idea muy aproximada de la diferencia entre dos colores.

Se considera que la diferencia entre dos colores empieza a ser perceptible a partir de un valor $\Delta E = 2{,}3$. Para un cálculo más preciso se recomienda la definición de $\Delta E2000$.

Para medir las diferencias de color entre los parches originales y los parches reconstruídos, en el presente trabajo se empleará la medida de la diferencia de color $\Delta E2000$ proporcionada por el software ColorLab excepto para el cálculo de la proximidad en el espacio CIELab entre dos colores en la sección 2.5.

Para calcular la diferencia $\Delta E2000$ entre dos cartas de color[18], se abren ambas cartas en File→ Open. Posteriormente se convierten a valores

[17] Definición propuesta por la Comisión Internacional de la Iluminación en 1976.

[18] Previamente será necesario haber trasladado toda la información procesada en MATLAB a varios ficheros .txt de ColorLab como se indica en la sección 2.1.11.

CIE en Filter → Conversion → CIE Colors (realizar esta operación dos veces, una para cada carta de color). Después se selecciona Special → Comparing y ColorLab genera un informe de las diferencias de color entre los parches de cada carta y un gráfico que representa estas diferencias haciendo click sobre la opción Save Report....

Error RMS

Sean $R_{\lambda O}$ y $R_{\lambda R}$ las curvas de reflectancia espectral en el dominio de la longitud de onda de los parches originales y reconstruídos respectivamente y sea n el número de longitudes de onda en las que se realiza el muestreo. Se define el error cuadrático medio E_{RMS} como:

$$E_{RMS} = \sqrt{\frac{1}{n}\sum_{\lambda_{i=1}}^{n} |R_{\lambda O} - R_{\lambda R}|^2} \tag{2.11}$$

A continuación se detalla una función de MATLAB para realizar el cálculo de dicho error:

```
function [error]=rms(A, B)

% rms devuelve una matriz de errores con los errores rms entre
% las filas de A y B. A y B deben tener las mismas dimensiones.
squareddif=0;

for i=1:size(A, 2)
    squareddif=squareddif+(A(1, i)-B(1, i))^2;
end
error=sqrt(squareddif/size(A, 2));

for j=2:size(A, 1)
    squareddif=0;
```

```
    for i=1:size(A, 2)
        squareddif=squareddif+(A(j, i)-B(j, i))^2;
    end
    error=[error; sqrt(squareddif/size(A, 2))];

end
end
```

2.1.8. Cálculo del valor metamérico

Uno de los objetivos que persigue el presente trabajo de investigación es reducir el metamerismo en la reconstrucción de un conjunto de parches de prueba. Por esta razón resulta necesario elegir un indicador del metamerismo de la reconstrucción y calcular su valor para cada uno de los parches reconstruidos con un iluminante (cálido y frío) y dos iluminantes para cada uno de los métodos empleados.

El indicador más adecuado a priori sería el índice de metamerismo. Este indicador se define como la diferencia ΔE que existe entre el parche de prueba y el original cuando estos son observados en condiciones distintas a aquellas bajo cuya observación coinciden[19]. Debido a su propia definición, para el cálculo del índice de metamerismo se exige que el parche de prueba y el original coincidan en determinadas circunstancias de iluminación y observación.

Llegados a este punto se presenta un problema y es no se tiene garantía de que el parche de prueba reconstruído y el original vayan a coincidir bajo iluminante y observador alguno. Este problema se puede salvar gracias a las aportaciones de Fairman [79]. Fairman menciona dos índices de metamerismo que se calculan de forma parecida. Uno de ellos es el índice específico de metamerísmo (debido a cambios en la iluminación) y otro es el índice general de metamerismo (debido a cambios en la iluminación y en el observador). Para calcular el índice

[19] Y su ΔE es nulo.

específico de metamerismo primero se halla N_c que es la distribución espectral corregida cuya diferencia de color con N_s bajo el iluminante considerado en el cálculo de la matriz R debe ser nula y cuya diferencia de color con N_s bajo el iluminante de prueba debe tomarse para el cálculo del índice de metamerismo para cambios en el iluminante:

$$N_c = R \times N_s + (I - R) * N_t \tag{2.12}$$

Donde N_s es la distribución espectral del parche de muestra (standard) y N_t es la distribución espectral del parche de prueba (trial). I es la matriz identidad y R es una matriz de tamaño 36x36 que se calcula como:

$$R = A \times (A' \times A)^{-1} \times A' \tag{2.13}$$

Siendo A la matriz de tamaño 36×3 que almacena la información característica del iluminante y del observador.

No obstante, en el presente trabajo de investigación se ha considerado una forma alternativa más simple de medir el efecto del metamerismo en la reconstrucción de los parches. En lugar de usar el índice de metamerismo, se compara el error ΔE cometido entre la reconstrucción y el original de cada parche de prueba bajo un iluminante A y bajo un iluminante D65 y su resta –en valor absoluto– dará un valor al que se ha llamado valor metamérico. Conviene tener en cuenta que las reconstrucciones que presentan un buen comportamiento metamérico frente a los cambios de iluminación, también lo hacen respecto al observador como indica Berns [80].

2.1.9. Cálculo del coeficiente de bondad del ajuste GFC

Para medir la bondad del ajuste espectral, además del valor RMS, se utiliza habitualmente el coeficiente de bondad de ajuste GFC que se calcula como:

$$GFC = \frac{\sum f f_r}{(\sum f^2)^{1/2} (\sum f_r^2)^{1/2}} \tag{2.14}$$

Donde f y f_r son las curvas de los parches originales y reconstruidas. Un *GFC* próximo a 1 indica una buena reconstrucción. Un GFC alejado de 1 indica una mala reconstrucción.

2.1.10. Convertir archivos .txt de ColorLab a matrices de MATLAB

Para poder realizar cálculos en MATLAB hay que trasladar la información de los ficheros .txt de ColorLab a MATLAB.

Para ello se ejectua una hoja de cálculo –como por ejemplo Microsoft Excel– y se abre el fichero .txt de ColorLab. Se selecciona la hoja entera de cálculo y en Editar→ Reemplazar, se reemplazan las comas por puntos.

A continuación Se seleccionan los valores numéricos correspondientes a las lecturas de los parches. Se copian en el portapapeles y se pegan en una nueva hoja de cálculo. Luego se guarda el fichero como una hoja .xls.

Se abre MATLAB, se selecciona Import data y se importa la hoja de calculo .xls como una matriz de MATLAB.

Ahora ya se dispone de los valores de color medidos con la cámara fotográfica y con el espectrofotómetro dentro de matrices que pueden utilizarse en MATLAB para realizar los cálculos pertinentes.

2.1.11. Convertir matrices de MATLAB a ficheros .txt de ColorLab

Se abre la matriz con ayuda del programa MATLAB y se selecciona en Edición → Copiar.

Con ayuda de la hoja de cálculo Excel, se abre un fichero .txt de ColorLab con el esqueleto de una medición de la carta de color (debe ser el esqueleto de una carta con las mismas dimensiones) que ya funcione, se sustituyen los valores actuales por los que se han copiado en el portapapeles procedentes de MATLAB y se guarda con otro nombre. Por supuesto es necesario reemplazar los puntos por comas previamente en Edición → Reemplazar. Y también al guardar el fichero hay que guardarlo

como documento de texto con valores delimitados por tabulaciones, para que este documento pueda posteriormente ser abierto con ColorLab.

2.1.12. Caracterización de los iluminantes empleados

Para que el experimento propuesto en el siguiente trabajo sea reproducible es preciso caracterizar de alguna manera los iluminantes empleados [81], [82], [83].

Por un lado se han empleado 4 iluminantes LED de la marca OSRAM modelo LED STAR CLASSIC A 60 de 10W de potencia, 810 lúmenes nominales, a 2700K, de rosca E27. Por otro lado se han empleado 4 iluminantes LED de la marca OSRAM modelo LED STAR CLASSIC A 60 de 10W de potencia, 810 lúmenes nominales, a 6500K, de rosca E27.

Cada grupo de 4 lámparas iguales se colocan de tal forma en el dispositivo que arrojen sin sombra alguna una iluminación uniforme de $20 cd/m^2$. Para que la caracterización de los iluminantes sea completa se ha medido además la curva de emitancia espectral de cada uno de ellos ajustada a un pico máximo de $20 cd/m^2$.

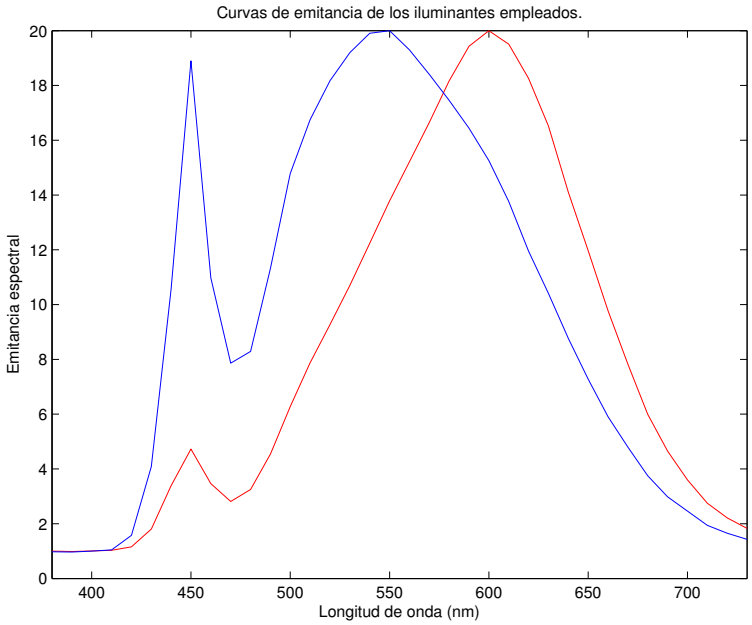

Figura 2.6: Curvas de emitancia espectral de los dos tipos de iluminantes utilizados. En rojo el iluminante a 2700K y en azul el iluminante a 6500K. Prestar atención al pico característico de las bombillas con tecnología LED.

Estas curvas de emitancia espectral se han medido con ayuda del espectrofotómetro HR4000 de Ocean Optics (distribuido por BFI OPTILAS) con el software de Ocean Optics Spectrasuite. Para realizar la medición se utiliza un filtro de densidad neutro con el objetigo de disminuir la intensidad luminosa que podría saturar el dispositivo. Los valores obtenidos de distribución espectral de energía son relativos y se han escalado de tal modo que el máximo se alcanze a $20 cd/m^2$.

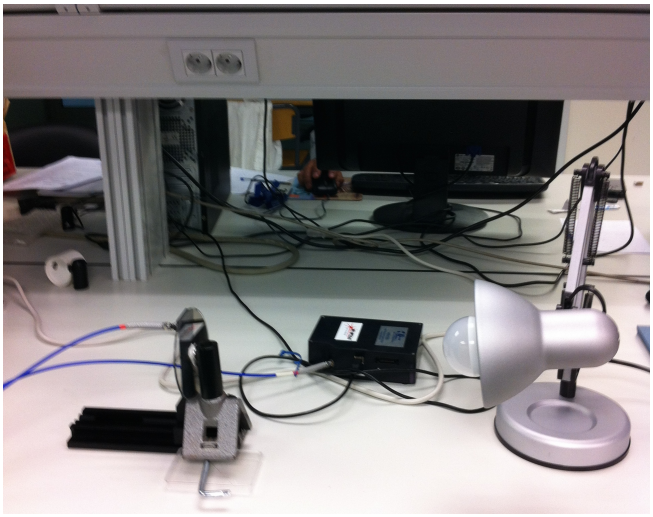

Figura 2.7: Primera fotografía del dispositivo empleado para medir el espectro de luz.

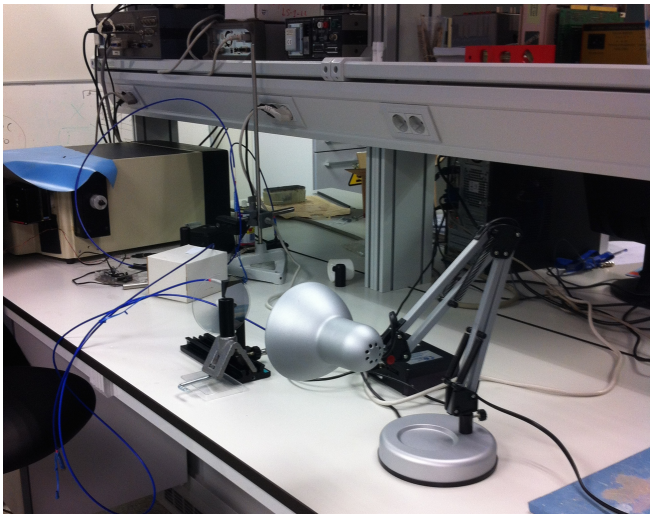

Figura 2.8: Segunda fotografía del dispositivo empleado para medir el espectro de luz.

2.1.13. Caracterización de los parches de color empleados

Las curvas de reflectancia espectral de los parches de color empleados para el entrenamiento y el control quedan representadas en las siguientes gráficas:

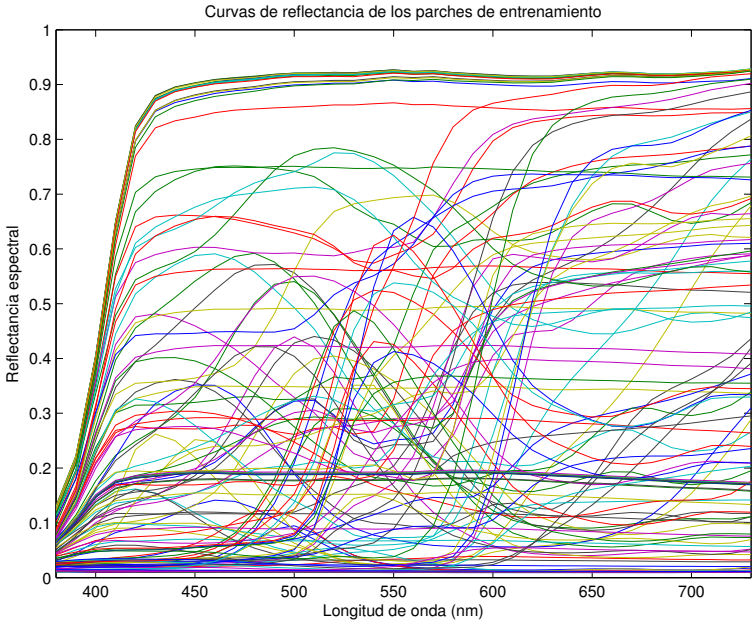

Figura 2.9: Curvas de reflectancia espectral de los parches de entrenamiento empleados.

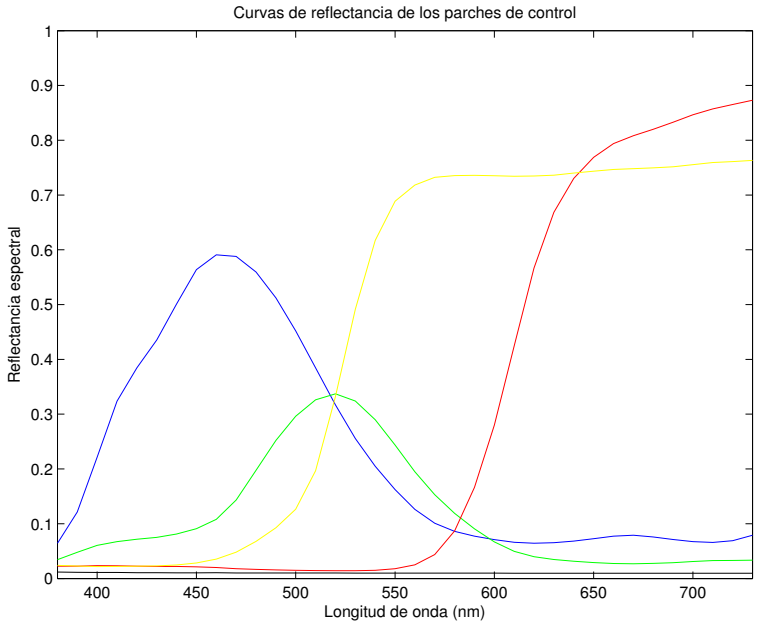

Figura 2.10: Curvas de reflectancia espectral de los parches de control empleados.

2.2. Método de la pseudoinversa directa (PINV)

Este método trata de encontrar una relación lineal entre los valores XYZ que proceden de la lectura RGB de los parches de la carta de entrenamiento y los valores espectrales de la carta de entrenamiento. Posteriormente dados unos valores XYZ de test podrán predecirse mediante esta relación lineal ya conocida las curvas espectrales de esos parches.

El problema que aquí se presenta consiste en que se está tratando de predecir para cada parche un conjunto de 36 valores correspondientes a sus longitudes de onda comprendidas entre los 380 y los 730 nanometros a partir de solo 3 variables XYZ (en el caso de realizar la reconstrucción con un solo ilumimante) o 6, lo que constituye un sistema compatible

indeterminado con más de una solución. De entre todas las posibles matrices inversas que podrían emplearse para solucionar el problema, la pseudoinversa de Moore–Penrose[20] es la que minimiza el error cuadrático medio y por ello resulta ideal si lo que se pretende es realizar una reconstrucción espectralmente fiel.

Antes de proceder a explicar las expresiones matemáticas que permiten realizar la reconstrucción con este método, se presentan y describen las variables que se utilizan.

`SPKTtraining` es una matriz de tamaño 135×36 que contiene las curvas de reflectancia espectral medidas con ayuda del espectrofotómetro. Cada fila corresponde a un parche de color de la carta de entrenamiento y cada columna corresponde a una longitud de onda entre los 380 y los 730 nanometros.

`XYZtraining2700`, `XYZtraining6500` y `XYZtraining` son las matrices de tamaño 135×3, 135×3 y 135×6 cuyas filas representan los parches de entrenamiento y en cuyas columnas se encuentran los valores XYZ de la carta de entrenamiento que proceden de los valores RGB medidos con la cámara bajo un iluminante a $2700K$, a $6500K$ y bajo ambos iluminantes respectivamente[21].

`X3ch2700`, `X3ch6500` y `X6ch` son las matrices de transformación que relacionan los valores XYZ para un iluminante a $2700K$, otro a $6500K$ y dos iluminantes ($2700K$ y $6500K$) respectivamente con los valores espectrales.

Para el caso de la reconstrucción con un iluminante a $2700K$ se tiene que la matriz de reflectancias espectrales de la carta de entrenamiento `SPKTtraining` es igual al producto de la matriz de valores XYZ `XYZtraining2700` multiplicada por la matriz de transformación `X3ch2700`:

$$\texttt{SPKTtraining} = \texttt{XYZtraining2700} \times \texttt{X3ch2700} \tag{2.15}$$

Así que queda clara la forma de calcular la matriz de transformación[22]:

[20] También llamada inversa generalizada.

[21] No simultáneamente sino las primeras columnas con el iluminante a $2700K$ y las tres siguientes con el iluminante a $6500K$

[22] pinv(...) es un comando de MATLAB que devuelve la pseudoinversa de Moore–

$$\texttt{X3ch2700} = \texttt{pinv(XYZtraining2700)} \times \texttt{SPKTtraining} \tag{2.16}$$

Una vez hallada la matriz de transformación, la reconstrucción espectral de los parches de test se realiza de forma análoga a 2.15, es decir:

$$\texttt{reconstructed3ch2700} = \texttt{XYZcontrol2700} \times \texttt{X3ch2700} \tag{2.17}$$

Aquí `XYZcontrol2700` es la matriz de tamaño 5×3 cuyas columnas almacenan los valores XYZ procedentes de la medida RGB a $2700K$ de cada uno de los parches de test ordenados por filas. `reconstructed3ch2700` es la matriz de tamaño 5×36 que contiene en sus columnas los valores de reflectancia espectral para cada uno de los parches de test reconstruídos y ordenados por filas.

La explicación que se acaba de detallar en los párrafos anteriores de esta sección para la reconstrucción con un iluminante a $2700K$ es extensible a la reconstrucción con un iluminantes a $6500K$ y con dos iluminantes.

A continuación se detalla el conjunto de instrucciones de MATLAB necesarias para reconstruir los parches de test y plotear las curvas de reflectancia espectral originales y reconstruidas de los parches de test de la carta ColorChecker SG por el método de la pseudoinversa directa [84]. También se incluyen instrucciones para el cálculo de los errores RMS cometidos durante la reconstrucción y para plotear un gráfico de barras en el que se comparan los errores RMS en cada uno de los parches con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes.

Los datos de entrada que se deben tener preparados antes de empezar a introducir estas instrucciones de MATLAB son `SPKTtraining`, `XYZtraining2700`, `XYZtraining6500` y `XYZtraining`. Los

Penrose. Si A es una matriz de tamaño $m \times n$ y $rg(A) = n$ entonces, la forma de calcular "manualmente"la matriz pseudoinversa es $\left(A^T A\right)^{-1} A^T$.

datos de salida que proporciona el método son `reconstructed3ch2700`, `reconstructed3ch6500`, `reconstructed6ch`, `RMS` y `deltaE`. El código de implementación de este método se encuentra en **??**.

2.3. Método de la descomposición en valores singulares (SVD)

Este método parte de la factorización de la matriz de reflectancias espectrales de entrenamiento `SPKTtraining`[23] de tamaño 135×36 en tres matrices `U` (135×3, reconstrucción con 1 iluminante) ó (135×6, reconstrucción con 2 iluminantes), `S` (3×3, reconstrucción con 1 iluminante) ó (6×6, reconstrucción con 2 iluminantes) y `V` (36×3, reconstrucción con 1 iluminante) ó (36×6, reconstrucción con 2 iluminantes). La matriz `S` es diagonal y contiene en orden de mayor a menor las longitudes de los ejes ortogonales del nuevo espacio que se utiliza como referencia para realizar la reconstrucción. Los coeficientes de cada parche en cada uno de estos ejes quedan definidos por `U` y `V` es la matriz que devuelve al nuevo espacio la forma original del espacio en el que se encuentran los parches originales de tal forma que:

$$\texttt{SPKTtraining} = \texttt{U} \times \texttt{S} \times \texttt{V'} \tag{2.18}$$

Para obtener `U`, `S` y `V` se utiliza el comando:

```
[U, S, V]=svds(SPKTtraining, 6);
```
(2.19)

Una vez calculados `U`, `S` y `V` se trata de encontrar una relación lineal entre los valores XYZ que proceden de la lectura RGB de los parches de la carta de entrenamiento y los coeficientes contenidos en la matriz `U`. Posteriormente dados unos valores XYZ de test podrán predecirse

[23] Contiene las curvas de reflectancia espectral medidas con ayuda del espectrofotómetro. Cada fila corresponde a un parche de color de la carta de entrenamiento y cada columna corresponde a una longitud de onda entre los 380 y los 730 nanometros.

mediante esta relación lineal ya conocida las curvas espectrales de esos parches.

Antes de proceder a explicar las expresiones matemáticas que permiten realizar la reconstrucción con este método, se presentan y describen las variables que se utilizan.

`XYZtraining2700`, `XYZtraining6500` y `XYZtraining` son las matrices de tamaño 135×3, 135×3 y 135×6 cuyas filas representan los parches de entrenamiento y en cuyas columnas se encuentran los valores XYZ de la carta de entrenamiento que proceden de los valores RGB medidos con la cámara bajo un iluminante a $2700K$, a $6500K$ y bajo ambos iluminantes respectivamente[24].

`X3ch2700`, `X3ch6500` y `X6ch` son las matrices de transformación que relacionan los valores XYZ para un iluminante a $2700K$, otro a $6500K$ y dos iluminantes ($2700K$ y $6500K$) respectivamente con los coeficientes contenidos en `U`.

Para el caso de la reconstrucción con un iluminante a $2700K$ se tiene que la matriz de coeficientes `U` es igual al producto de la matriz de valores XYZ `XYZtraining2700` multiplicada por la matriz de transformación `X3ch2700`:

$$\texttt{U} = \texttt{XYZtraining2700} \times \texttt{X3ch2700} \tag{2.20}$$

Así que queda clara la forma de calcular la matriz de transformación[25]:

$$\texttt{X3ch2700} = \texttt{pinv(XYZtraining2700)} \times \texttt{U} \tag{2.21}$$

Una vez hallada la matriz de transformación, la reconstrucción espectral de los parches de test se realiza a partir de 2.20 y 2.18, es decir:

[24] No simultáneamente sino las primeras columnas con el iluminante a $2700K$ y las tres siguientes con el iluminante a $6500K$

[25] pinv(...) es un comando de MATLAB que devuelve la pseudoinversa de Moore–Penrose. Si A es una matriz de tamaño $m \times n$ y $rg(A) = n$ entonces, la forma de calcular "manualmente"la matriz pseudoinversa es $\left(A^T A\right)^{-1} A^T$.

$$\texttt{reconstructed3ch2700} = \texttt{XYZcontrol2700} \times \texttt{X3ch2700} \times \texttt{S} \times \texttt{V'} \quad (2.22)$$

Aquí `XYZcontrol2700` es la matriz de tamaño 5×3 cuyas columnas almacenan los valores XYZ procedentes de la medida RGB a $2700K$ de cada uno de los parches de test ordenados por filas. `reconstructed3ch2700` es la matriz de tamaño 5×36 que contiene en sus columnas los valores de reflectancia espectral para cada uno de los parches de test reconstruídos y ordenados por filas.

La explicación que se acaba de detallar en los párrafos anteriores de esta sección para la reconstrucción con un iluminante a $2700K$ es extensible a la reconstrucción con un iluminantes a $6500K$ y con dos iluminantes.

El método SVD debe producir los mismos resultados que el método PINV ya que su fundamento teórico el mismo. Simplemente se incluye en el presente trabajo a modo de comprobación.

A continuación se detalla el conjunto de instrucciones de MATLAB necesarias para reconstruir los parches de test y plotear las curvas de reflectancia espectral originales y reconstruidas de los parches de test de la carta ColorChecker SG por el método de la descomposición en valores singulares[26]. También se incluyen instrucciones para el cálculo de los errores RMS cometidos durante la reconstrucción y para plotear un gráfico de barras en el que se comparan los errores RMS en cada uno de los parches con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes.

Los datos de entrada que se deben tener preparados antes de empezar a introducir estas instrucciones de MATLAB son `SPKTtraining`, `XYZtraining2700`, `XYZtraining6500` y `XYZtraining`. Los

[26]Previamente se intentó realizar una descomposición en componentes principales, el problema es que la predicción de la reflectancia espectral de los parches con este método es harto complicada porque el análisis elimina los valores medios de cada canal antes de hacer la descomposición y por eso luego no es posible realizar una reconstrucción exacta. Así que finalmente se optó por realizar una descomposición en valores singulares.

datos de salida que proporciona el método son `reconstructed3ch2700`, `reconstructed3ch6500`, `reconstructed6ch`, `RMS` y `deltaE`. El código de implementación de este método se encuentra en **??**.

2.4. Método del análisis de componentes independientes (ICA)

El análisis de componentes independientes (ICA) es un modelo apropiado para el uso de observaciones multivariadas. Supone que existen unas mezclas lineales de determinadas variables latentes a las que llega haciendo uso de la transformada discreta de Fourier cuya suma produce la muestra que está siendo observada. Sobre dichas variables latentes no se tiene –a priori– información alguna ni tampoco sobre la manera en que éstas se combinan. Dichas variables latentes–también conocidas como fuentes o factores– se asumen como no gaussianas y mutuamente independientes [27]. ICA decorrelaciona los datos y los descompone como suma ponderada de unas variables latentes que se suponen independientes y no gaussianas y sobre las cuales no se tiene información alguna a priori.

Se espera que el método ICA, en general, arroje resultados diferentes a PINV. Una diferencia notable entre los resultados con ICA y con PINV a favor de PINV significaría que la información contenida en la carta de color no puede ser explicada con un modelo basado en una suma de determinados colores base (o señales latentes) mejor que con un modelo basado en una relación lineal directa entre las respuestas triestímulo y las curvas espectrales, y constituiría una prueba positiva de la calidad de la carta de color empleada. Así pues, el método ICA empleado en el presente trabajo tiene una doble justificación: la de método que por sí mismo puede servir para la reconstrucción de curvas espectrales pudiéndose utilizar para comparar resultados y la de indicador de la calidad de la carta de color empleada.

[27] Esto quiere decir que el valor o los valores de una variable no proporcionan información alguna sobre el valor del resto de variables.

Por esta razón cabría esperar que si la carta de color de entrenamiento hubiera sido generada por una mezcla de seis colores, que fueran esos seis colores las variables latentes obtenidas por este método.

Antes de proceder a explicar las expresiones matemáticas que permiten realizar la reconstrucción con este método, se presentan y describen las variables que se utilizan.

`SPKTtraining` es una matriz de tamaño 135×36 que contiene las curvas de reflectancia espectral medidas con ayuda del espectrofotómetro. Cada fila corresponde a un parche de color de la carta de entrenamiento y cada columna corresponde a una longitud de onda entre los 380 y los 730 nanometros.

Para realizar el análisis[28]:

```
[icasig, mix, W] = fastica(SPKTtraining, 'lastEig', 3)   (2.23)
```

Opcionalmente se pueden añadir los parámetros `'maxNumIterations', 100000` para ampliar el número de iteraciones y garantizar la obtención del resultado. `icasig` (3×36, un iluminante) en este caso, debido al tercer parámetro de la función, guarda tres variables latentes y mix (135×3, un iluminante) es una matriz de coeficientes que almacena la forma en que se combinan esas tres variables latentes para dar la muestra.

Para hacer la reconstrucción con dos iluminantes sería:

```
[icasig, mix, W] = fastica(SPKTtraining, 'lastEig', 6)   (2.24)
```

y entonces `icasig` tendría como tamaño (6×36) y mix (135×6).

`XYZtraining2700`, `XYZtraining6500` y `XYZtraining` son las matrices de tamaño 135×3, 135×3 y 135×6 cuyas filas representan los parches de entrenamiento y en cuyas columnas se encuentran los valores XYZ de la carta de entrenamiento que proceden de los valores RGB medidos con

[28] Es preciso antes haber importado FastICA en MATLAB.

la cámara bajo un iluminante a 2700K, a 6500K y bajo ambos iluminantes respectivamente[29].

`X3ch2700`, `X3ch6500` y `X6ch` son las matrices de transformación que relacionan los valores XYZ para un iluminante a 2700K, otro a 6500K y dos iluminantes (2700K y 6500K) respectivamente con los coeficientes de mezcla `mix` obtenidos del análisis ICA.

Para el caso de la reconstrucción con un iluminante a 2700K se tiene que la matriz de coeficientes de mezcla `mix` es igual al producto de la matriz de valores XYZ `XYZtraining2700` multiplicada por la matriz de transformación `X3ch2700`:

$$\texttt{mix} = \texttt{XYZtraining2700} \times \texttt{X3ch2700} \tag{2.25}$$

Así que queda clara la forma de calcular la matriz de transformación[30]:

$$\texttt{X3ch2700} = \texttt{pinv(XYZtraining2700)} \times \texttt{mix} \tag{2.26}$$

Una vez hallada la matriz de transformación, la reconstrucción espectral de los parches de test se realiza teniendo en cuenta 2.25:

$$\texttt{reconstructed3ch2700} = \texttt{XYZcontrol2700} \times \texttt{X3ch2700} \times \texttt{icasig} \tag{2.27}$$

Aquí `XYZcontrol2700` es la matriz de tamaño 5×3 cuyas columnas almacenan los valores XYZ procedentes de la medida RGB a 2700K de cada uno de los parches de test ordenados por filas. `reconstructed3ch2700` es la matriz de tamaño 5×36 que contiene en sus columnas los valores de reflectancia espectral para cada uno de los parches de test reconstruídos y ordenados por filas.

La explicación que se acaba de detallar en los párrafos anteriores

[29] No simultáneamente sino las primeras columnas con el iluminante a 2700K y las tres siguientes con el iluminante a 6500K

[30] pinv(...) es un comando de MATLAB que devuelve la pseudoinversa de Moore–Penrose. Si A es una matriz de tamaño $m \times n$ y $rg(A) = n$ entonces, la forma de calcular "manualmente"la matriz pseudoinversa es $\left(A^T A\right)^{-1} A^T$.

de esta sección para la reconstrucción con un iluminante a 2700K es extensible a la reconstrucción con un iluminantes a 6500K y con dos iluminantes.

A continuación se detalla el conjunto de instrucciones de MATLAB necesarias para reconstruir los parches de test y plotear las curvas de reflectancia espectral originales y reconstruidas de los parches de test de la carta ColorChecker SG por el método del análisis de componentes independientes[31]. También se incluyen instrucciones para el cálculo de los errores RMS cometidos durante la reconstrucción y para plotear un gráfico de barras en el que se comparan los errores RMS en cada uno de los parches con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes.

Los datos de entrada que se deben tener preparados antes de empezar a introducir estas instrucciones de MATLAB son `SPKTraining`, `XYZtraining2700`, `XYZtraining6500` y `XYZtraining`. Los datos de salida que proporciona el método son `reconstructed3ch2700`, `reconstructed3ch6500`, `reconstructed6ch`, `RMS` y `deltaE`. El código de implementación de este método se encuentra en **??**.

2.5. Método de los colores cercanos (CC)

Este método consiste en elegir un número determinado de parches de color que existen en la carta de entrenamiento[32] de tal modo que su distancia euclídea en el espacio Lab a los parches que se desea reconstruir sea mínima. A estos parches se les da el nombre de parches de referencia.

Al considerar la mínima distancia euclídea en el espacio Lab cabe esperar de este método que sus resultados sean buenos o especialmente buenos cuando los parches de entrenamiento sean poco metaméricos respecto a los parches que se desea reconstruir[33].

[31] Este tipo de análisis descompone una señal en una mezcla de subcomponentes aditivos independientes. para el presente trabajo se ha utilizado la implementación en MATLAB de FastICA (`http://research.ics.aalto.fi/ica/fastica/`).

[32] Cuyos valores XYZ y espectrales son conocidos.

[33] Como en el caso de que usen el mismo pigmento, por ejemplo.

Una de las preguntas que se plantean sería el número de colores cercanos (es decir el valor de la variable `nParchesC`) a elegir para realizar la reconstrucción con el menor error RMS[34] posible. Para responder a esta pregunta y explorar los límites de este método se ha implementado la función `metodo_CC` que se utiliza para realizar sucesivas reconstrucciones empleando `nParchesC` colores de referencia en una simulación de Monte-Carlo. La llamada a esta función se realiza sucesivas veces, con distintos valores de `nParchesC`. `metodo_CC` devuelve tres matrices columna, una para los errores cometidos en la reconstrucción de los parches de test empleando el método de los colores cercanos con un iluminante a 2700K, otra para los errores cometidos empleando dicho método con un iluminante a 6500K y otra para los errores cometidos con dos iluminantes. Una vez conocido este valor, se procede a la reconstrucción espectral.

Para el caso de la reconstrucción con un sólo iluminante a 2700K, primero se calcula una matriz `dist2700` que contiene la distancia en el espacio Lab desde el parche de test `i` al parche de entrenamiento `j` donde `i` hace referencia a las filas de `dist2700` y `j` hace referencia a las columnas.

Posteriormente se almacenan los valores XYZ de los parches de entrenamiento que se utilizarán como referencia (`XYZtrainingA2`, `XYZtrainingB5`, `XYZtrainingD9`, `XYZtrainingL3`, `XYZtrainingM7`) y los valores espectrales de los parches de entrenamiento que se utilizarán como referencia (`SPKTtrainingA2`, `SPKTtrainingB5`, `SPKTtrainingD9`, `SPKTtrainingL3`, `SPKTtrainingM7`).

Con todas estas matrices de datos espectrales y datos XYZ de los parches de entrenamiento que se utilizarán como referencia, se considera que entre ellos hay una relación lineal según `X3ch2700A2`, `X3ch2700B5`, `X3ch2700D9`, `X3ch2700L3`, `X3ch2700M7` que se obtienen como:

[34]Se elige el error RMS y no el error $\Delta E2000$ porque lo que se persigue es una buena reconstrucción espectral.

$$\texttt{X3ch2700A2} = \texttt{pinv(XYZtraining2700A2)} \times \texttt{SPKTtrainingA2} \quad (2.28)$$

$$\texttt{X3ch2700B5} = \texttt{pinv(XYZtraining2700B5)} \times \texttt{SPKTtrainingB5} \quad (2.29)$$

$$\texttt{X3ch2700D9} = \texttt{pinv(XYZtraining2700D9)} \times \texttt{SPKTtrainingD9} \quad (2.30)$$

$$\texttt{X3ch2700L3} = \texttt{pinv(XYZtraining2700L3)} \times \texttt{SPKTtrainingL3} \quad (2.31)$$

$$\texttt{X3ch2700M7} = \texttt{pinv(XYZtraining2700M7)} \times \texttt{SPKTtrainingM7} \quad (2.32)$$

Y la reconstrucción de los parches queda como sigue:

$$\texttt{rA23ch2700} = \texttt{XYZcontrol2700(A2, :)} \times \texttt{X3ch2700A2} \quad (2.33)$$

$$\texttt{rB53ch2700} = \texttt{XYZcontrol2700(B5, :)} \times \texttt{X3ch2700B5} \quad (2.34)$$

$$\texttt{rD93ch2700} = \texttt{XYZcontrol2700(D9, :)} \times \texttt{X3ch2700D9} \quad (2.35)$$

$$\texttt{rL33ch2700} = \texttt{XYZcontrol2700(L3, :)} \times \texttt{X3ch2700L3} \quad (2.36)$$

$$\texttt{rM73ch2700} = \texttt{XYZcontrol2700(M7, :)} \times \texttt{X3ch2700M7} \quad (2.37)$$

El código de implementación de este método se encuentra en **??**.

2.6. Método cuadrático de los colores cercanos (CCC)

El método cruadrático de los colores cercanos (CCC) es una versión mejorada del método de los colores cercanos. La única diferencia entre el método de los colores cercanos y el método cuadrático de los colores cercanos es que en este segundo método en lugar de únicamente tomarse los valores XYZ de los parches de entrenamiento para entrenar el sistema y los valores XYZ de los parches de test para realizar la predicción de las curvas de reflectancia espectral de los parches de test, se emplean adicionalmente los productos cruzados y los cuadrados de las componentes XYZ [57], [58].

Para ello resulta de gran utilidad la función cuadratizar que trans-

forma una matriz de valores XYZ ordenados en filas en otra matriz que contiene 3 columnas adicionales con los productos cruzados y 3 columnas más con los cuadrados de los valores XYZ.

```
function [X]=cuadratizar(Y);

X=[Y, (Y(:, 1).*Y(:, 2)), (Y(:, 1).*Y(:, 3)),...
    (Y(:, 2).*Y(:, 3)), (Y(:, 1).*Y(:, 1)),...
    (Y(:, 2).*Y(:, 2)), (Y(:, 3).*Y(:, 3))];

end
```

En esta ocasión, al igual que se hizo en la sección 2.5 resulta oportuno plantearse qué número de parches de referencia conviene tomar para minimizar la suma de los errores RMS cometidos en la reconstrucción de las curvas de reflectancia espectral con un iluminante a 2700K, con un iluminante a 6500K y con dos iluminantes.

Para ello emplearemos el código de la sección **??** que se parece mucho al de **??**.

Capítulo 3

Resultados

En el presente capítulo se muestran las gráficas y tablas correspondientes a los resultados obtenidos con los distintos métodos de reconstrucción estudiados que son: método de la pseudoinversa directa, método de la descomposición en valores singulares, método del análisis de componentes independientes, método de los colores cercanos y método cuadrático de los colores cercanos.

3.1. Método de la pseudoinversa directa (PINV)

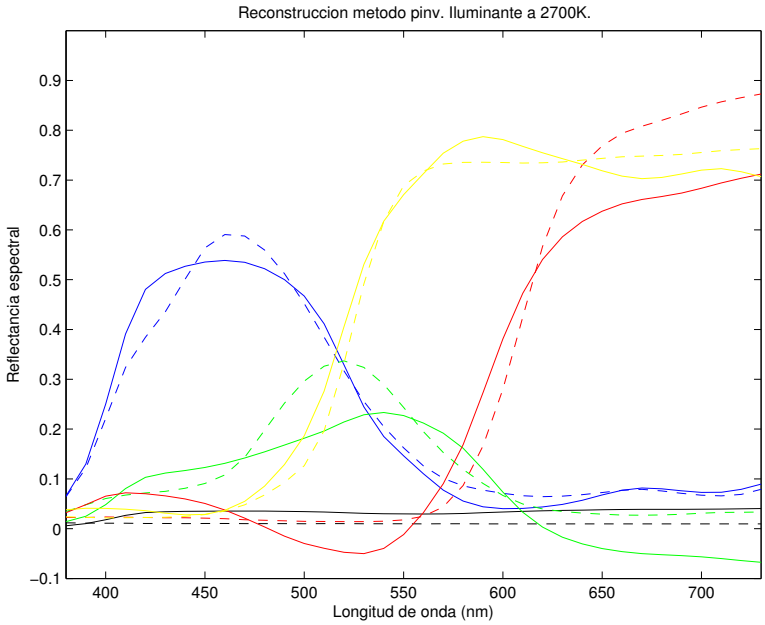

Figura 3.1: Reconstrucción de los 5 parches de test realizada con el método de la pseudoinversa empleando 1 único iluminante a 2700K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

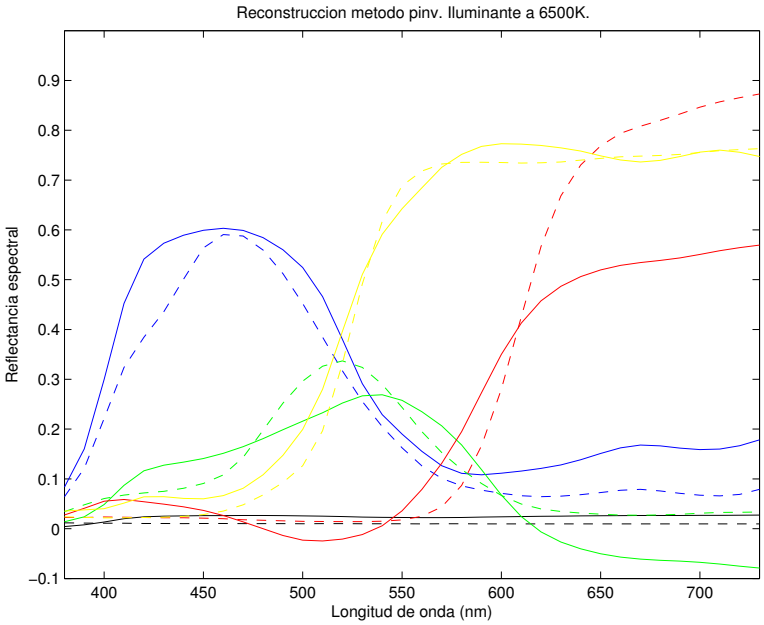

Figura 3.2: Reconstrucción de los 5 parches de test realizada con el método de la pseudoinversa empleando 1 único iluminante a 6500K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

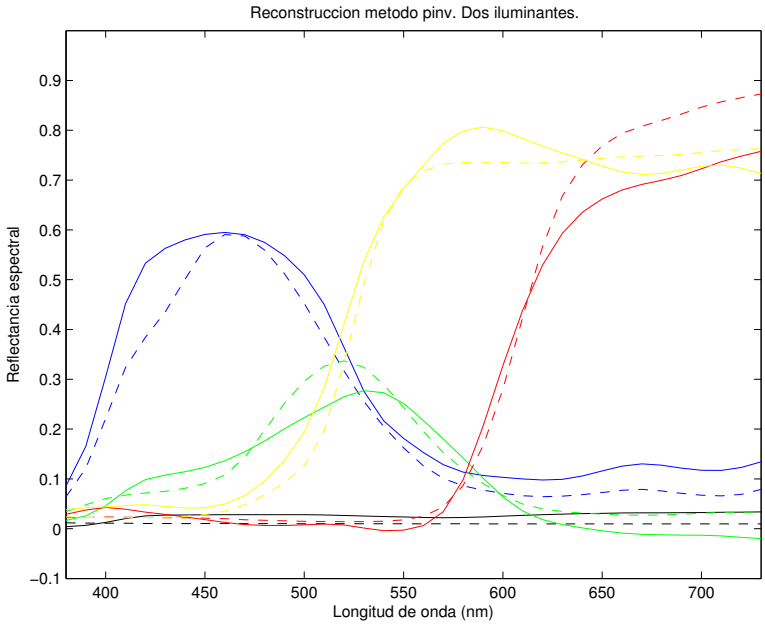

Figura 3.3: Reconstrucción de los 5 parches de test realizada con el método de la pseudoinversa empleando 2 iluminantes distintos (6 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.0241/0.9644	0.0146/0.9685	0.0176/0.9582
Azul B5	0.0318/0.9941	0.0730/0.9866	0.0567/09919
Verde D9	0.0647/0.9073	0.0636/0.9094	0.0372/0.9717
Rojo L3	0.0905/0.9908	0.1558/0.9840	0.0640/0.9982
Amarillo M7	0.0351/0.9980	0.0340/0.9982	0.0394/0.9976
MEDIA	0.0492/0.9709	0.0682/0.9694	0.0429/0.9835
DESV. TIP.	0.0248/0.0339	0.0485/0.0314	0.0163/0.0159

Tabla 3.1: Errores *RMS* y coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la pseudoinversa.

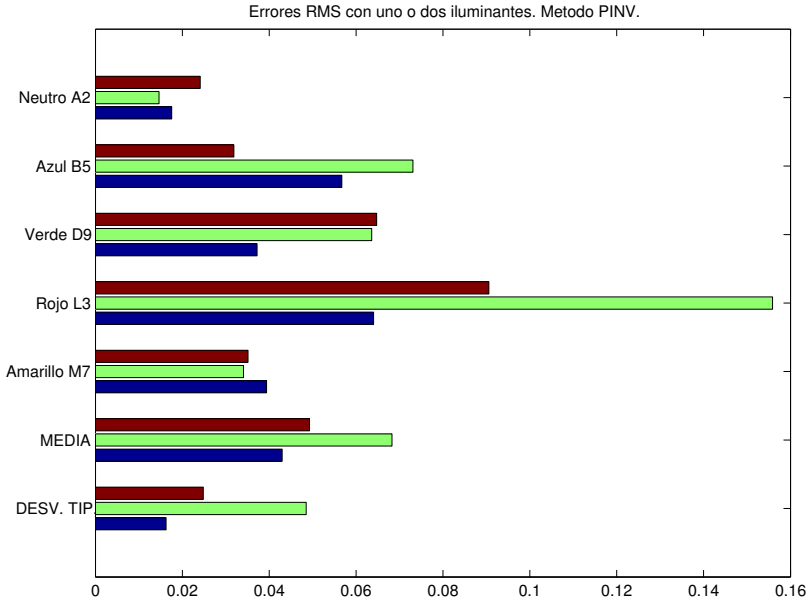

Figura 3.4: Errores *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la pseudoinversa.

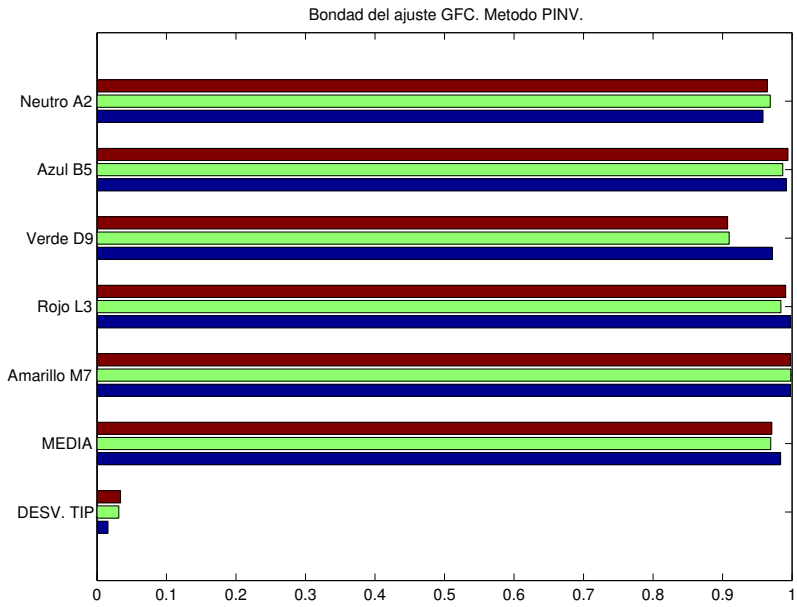

Figura 3.5: Coeficientes *GFC* de la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la pseudoinversa.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.61	5.74	6.45
Azul B5	2.67	5.26	3.93
Verde D9	5.87	4.59	3.51
Rojo L3	7.09	2.52	1.26
Amarillo M7	1.33	3.06	1.92
MEDIA	5.11	4.23	3.41
DESV. TIP.	2.72	1.24	1.80

Tabla 3.2: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la pseudoinversa. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

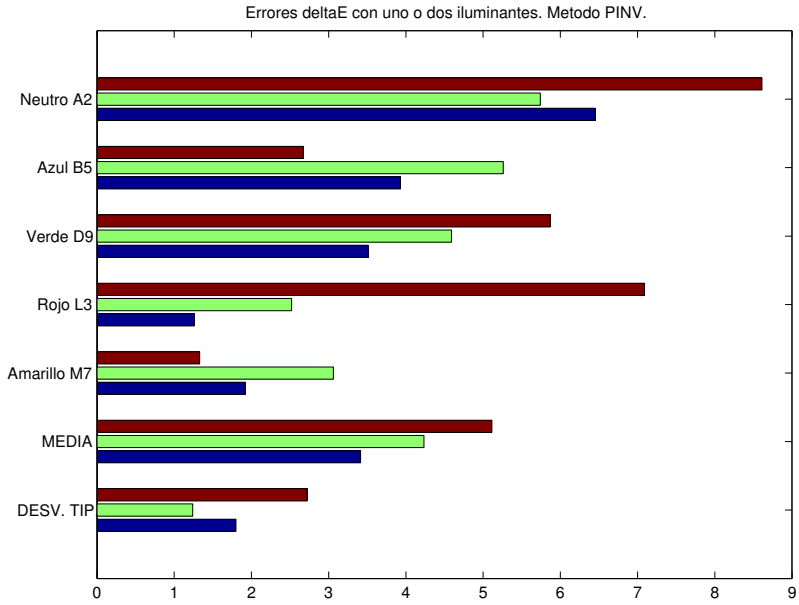

Figura 3.6: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la pseudoinversa. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	9.80	6.64	7.52
Azul B5	4.21	9.07	5.95
Verde D9	3.33	3.65	1.80
Rojo L3	2.40	4.08	1.30
Amarillo M7	0.91	2.47	1.42
MEDIA	4.13	5.18	3.60
DESV. TIP.	3.04	2.37	2.61

Tabla 3.3: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la pseudoinversa. El iluminante tomado como referencia para el cálculo del error perceptivo es el A (2856K).

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.64	5.89	6.51
Azul B5	2.29	4.78	3.66
Verde D9	7.44	6.23	4.67
Rojo L3	8.97	2.28	1.31
Amarillo M7	1.54	3.53	2.24
MEDIA	5.78	4.54	3.68
DESV. TIP.	3.20	1.47	1.83

Tabla 3.4: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la pseudoinversa. El iluminante tomado como referencia para el cálculo del error perceptivo es el D65.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	1.16	0.75	1.01
Azul B5	1.92	4.29	2.29
Verde D9	4.11	2.58	2.87
Rojo L3	6.57	1.80	0.01
Amarillo M7	0.63	1.06	0.82
MEDIA	2.88	2.10	1.40
DESV. TIP.	2.19	1.27	1.04

Tabla 3.5: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la pseudoinversa.

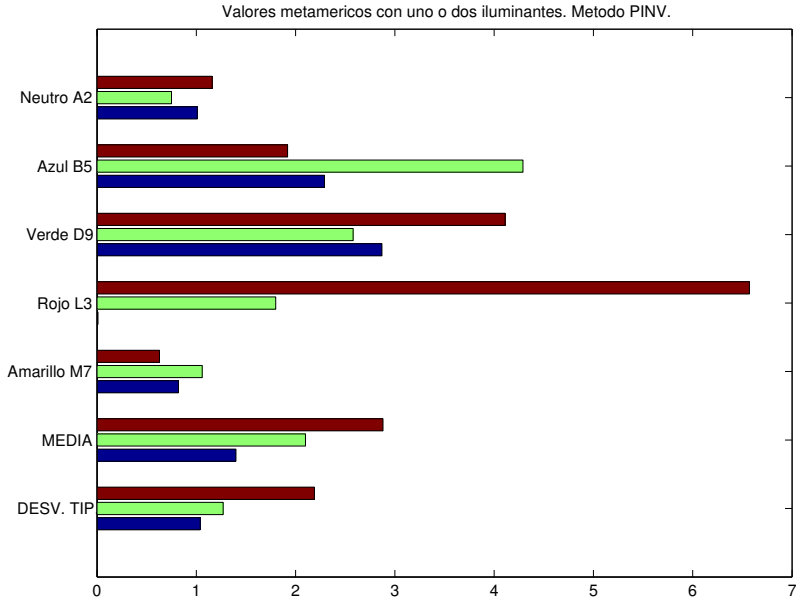

Figura 3.7: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la pseudoinversa.

3.2. Método de la descomposición en valores singulares (SVD)

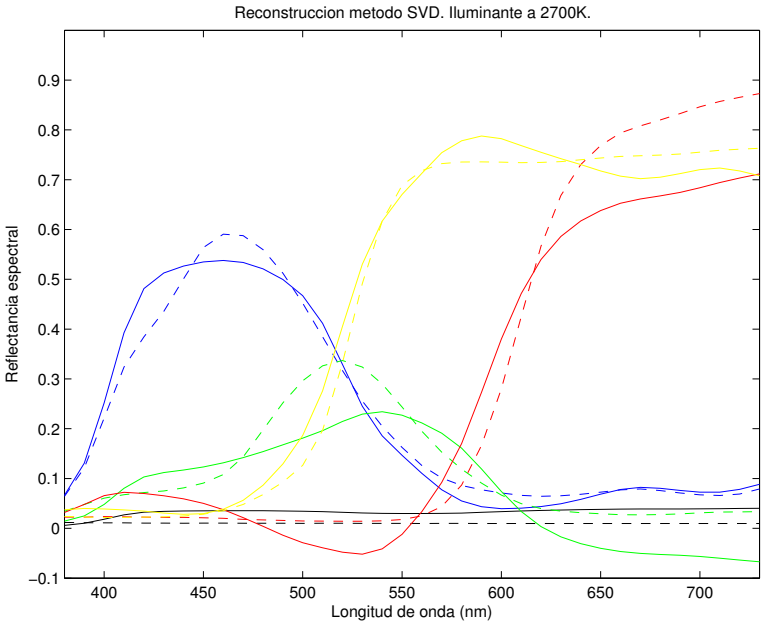

Figura 3.8: Reconstrucción de los 5 parches de test realizada con el método de la descomposición en valores singulares empleando 1 único iluminante a 2700K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

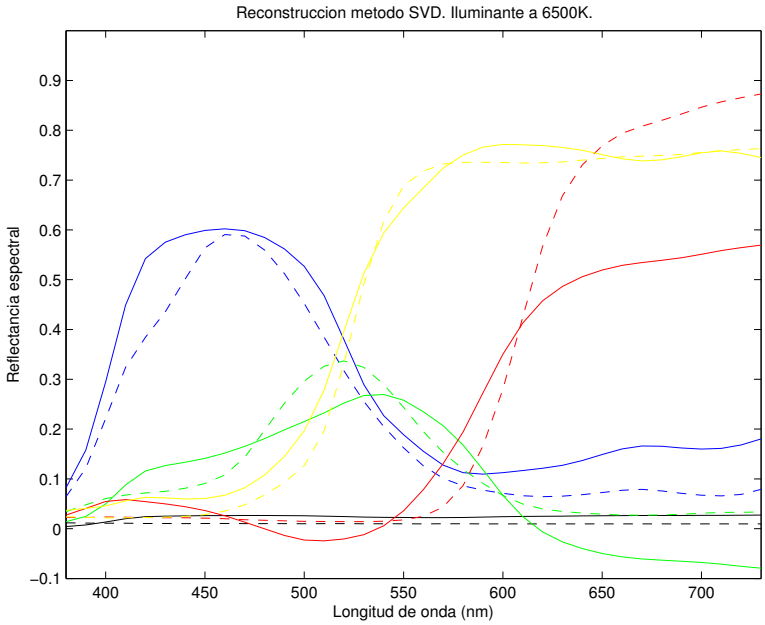

Figura 3.9: Reconstrucción de los 5 parches de test realizada con el método de la descomposición en valores singulares empleando 1 único iluminante a 6500K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

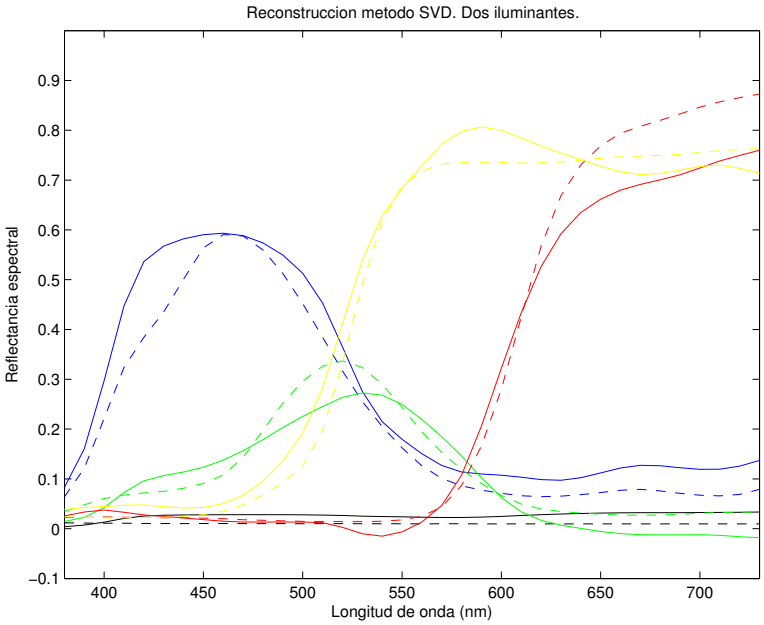

Figura 3.10: Reconstrucción de los 5 parches de test realizada con el método de la descomposición en valores singulares empleando 2 iluminantes distintos (6 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.0241/0.9645	0.0146/0.9684	0.0175/0.9582
Azul B5	0.0323/0.9939	0.0731/0.9866	0.0568/0.9918
Verde D9	0.0647/0.9073	0.0636/0.9094	0.0374/0.9713
Rojo L3	0.0905/0.9908	0.1559/0.9840	0.0637/0.9983
Amarillo M7	0.0351/0.9980	0.0337/0.9983	0.0394/0.9976
MEDIA	0.0493/0.9709	0.0682/0.9693	0.0430/0.9837
DESV. TIP.	0.0248/0.0339	0.0486/0.0314	0.0162/0.0157

Tabla 3.6: Errores *RMS* y coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la descomposición en valores singulares.

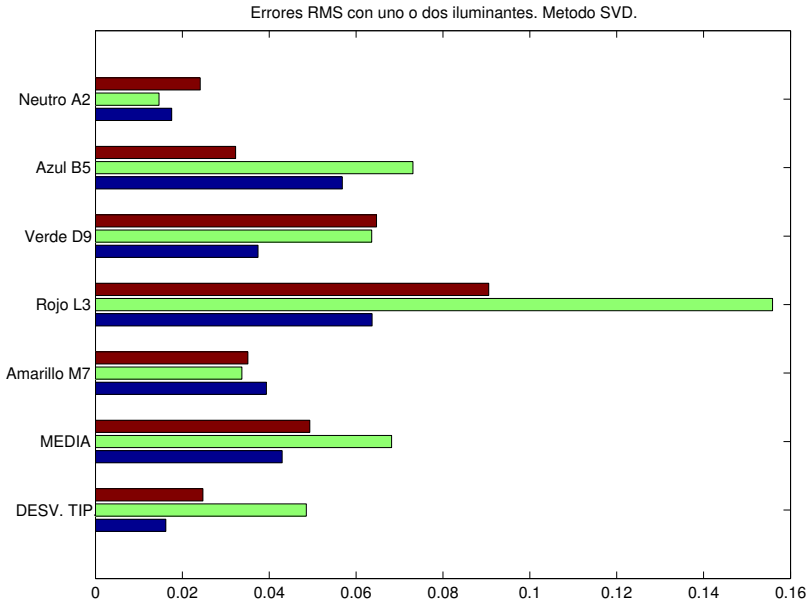

Figura 3.11: Errores *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la descomposición en valores singulares.

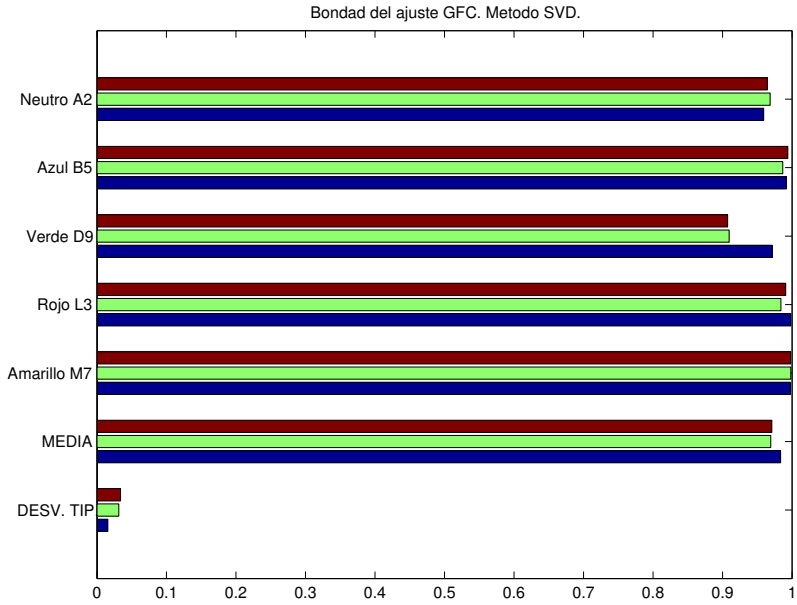

Figura 3.12: Coeficiente *GFC* de la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la descomposición en valores singulares.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.60	5.74	6.42
Azul B5	2.69	5.29	3.98
Verde D9	5.88	4.58	3.62
Rojo L3	7.07	2.53	1.37
Amarillo M7	1.34	3.02	1.93
MEDIA	5.11	4.23	3.46
DESV. TIP.	2.71	1.25	1.78

Tabla 3.7: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la descomposición en valores singulares. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

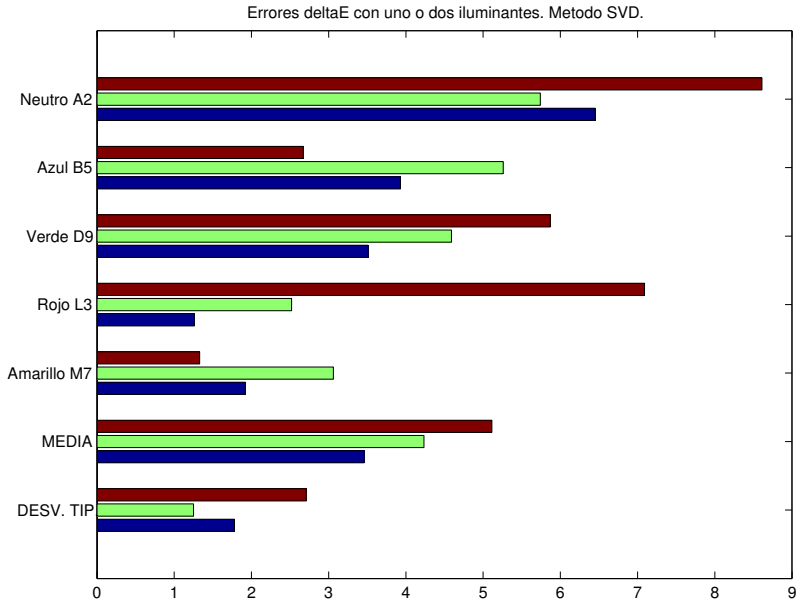

Figura 3.13: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la descomposición en valores singulares. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.63	6.64	7.49
Azul B5	2.31	9.14	6.07
Verde D9	7.46	3.65	1.86
Rojo L3	8.95	4.10	1.22
Amarillo M7	1.54	2.43	1.42
MEDIA	5.78	5.19	3.61
DESV. TIP.	3.19	2.40	2.63

Tabla 3.8: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la descomposición en valores singulares. El iluminante tomado como referencia para el cálculo del error perceptivo es el A (2856K).

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	2.92	5.89	6.49
Azul B5	7.10	4.80	3.71
Verde D9	3.13	6.22	4.80
Rojo L3	9.06	2.27	1.53
Amarillo M7	3.22	3.49	2.25
MEDIA	5.09	4.53	3.76
DESV. TIP.	2.52	1.48	1.78

Tabla 3.9: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la descomposición en valores singulares. El iluminante tomado como referencia para el cálculo del error perceptivo es el D65.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	1.17	0.75	1.00
Azul B5	1.92	4.34	2.36
Verde D9	4.13	2.57	2.94
Rojo L3	6.56	1.83	0.31
Amarillo M7	0.63	1.06	0.83
MEDIA	2.88	2.11	1.49
DESV. TIP.	2.19	1.28	0.99

Tabla 3.10: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de la descomposición en valores singulares.

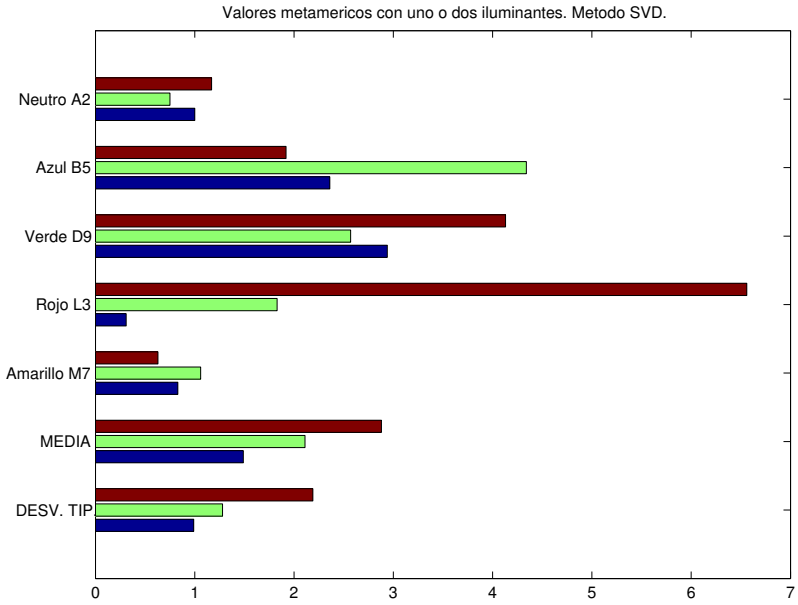

Figura 3.14: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de la descomposición en valores singulares.

3.3. Método del análisis de componentes independientes (ICA)

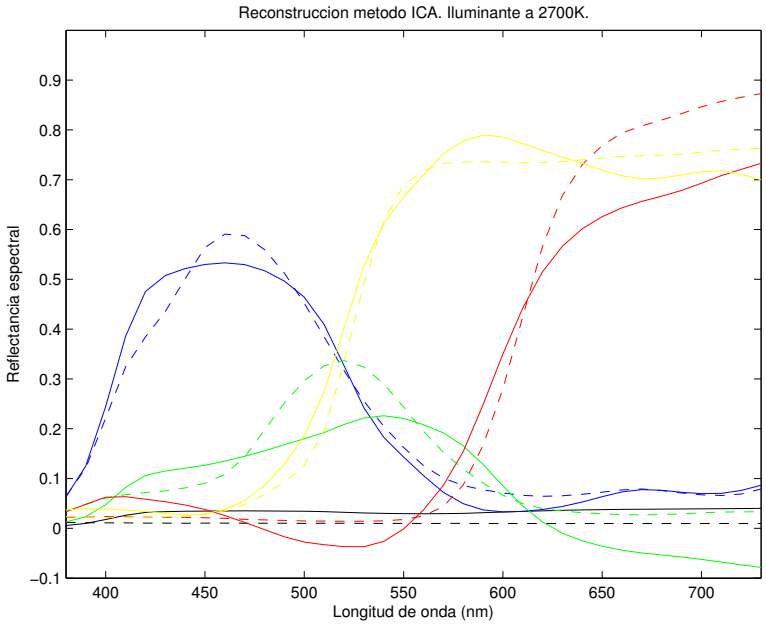

Figura 3.15: Reconstrucción de los 5 parches de test realizada con el método de análisis de componentes independientes empleando 1 único iluminante a 2700K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

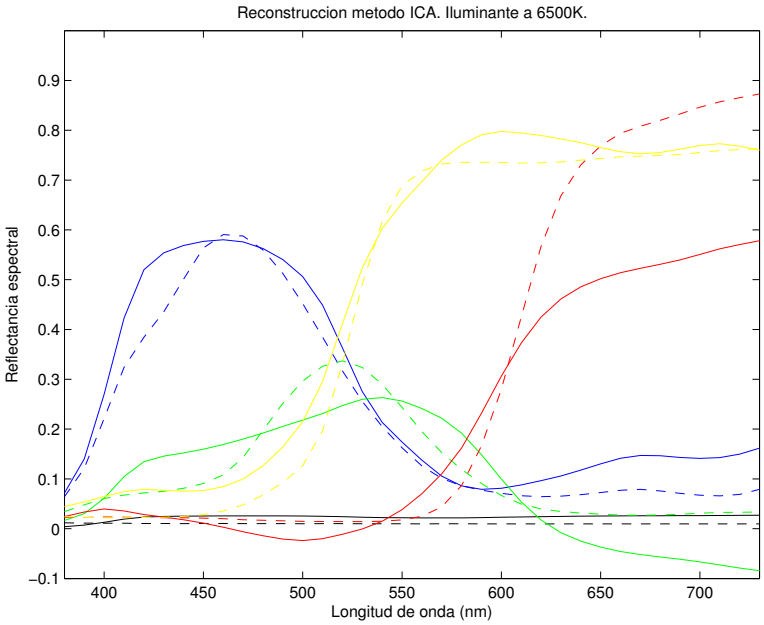

Figura 3.16: Reconstrucción de los 5 parches de test realizada con el método de análisis de componentes independientes empleando 1 único iluminante a 6500K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

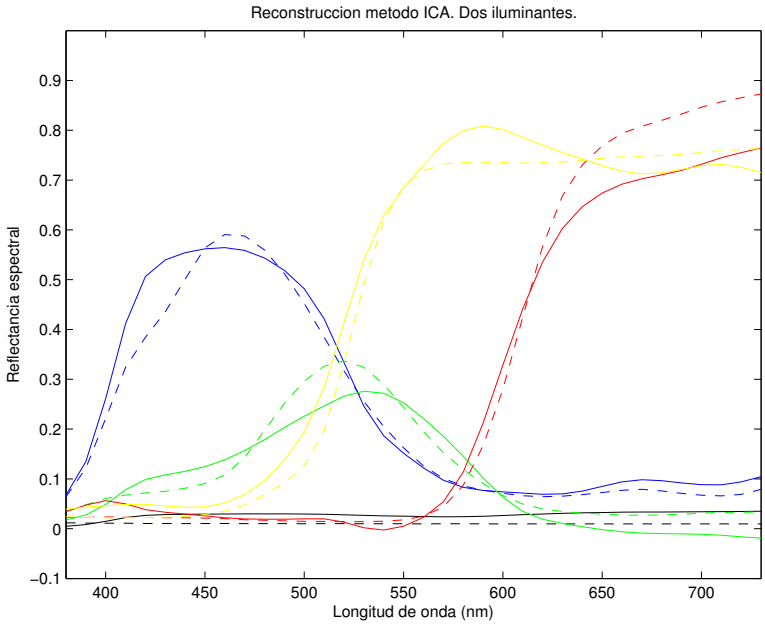

Figura 3.17: Reconstrucción de los 5 parches de test realizada con el método de análisis de componentes independientes empleando 2 iluminantes distintos (6 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.0237/0.9637	0.0141/0.9673	0.0189/0.9617
Azul B5	0.0326/0.9937	0.0561/0.9896	0.0361/0.9943
Verde D9	0.0673/0.8984	0.0659/0.9054	0.0365/0.9723
Rojo L3	0.0861/0.9940	0.1580/0.9906	0.0592/0.9981
Amarillo M7	0.0360/0.9979	0.0462/0.9973	0.0400/0.9976
MEDIA	0.0492/0.9696	0.0681/0.9700	0.0382/0.9848
DESV. TIP.	0.0236/0.0376	0.0482/0.0339	0.0129/0.0149

Tabla 3.11: Errores *RMS* y coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método del análisis de componentes independientes.

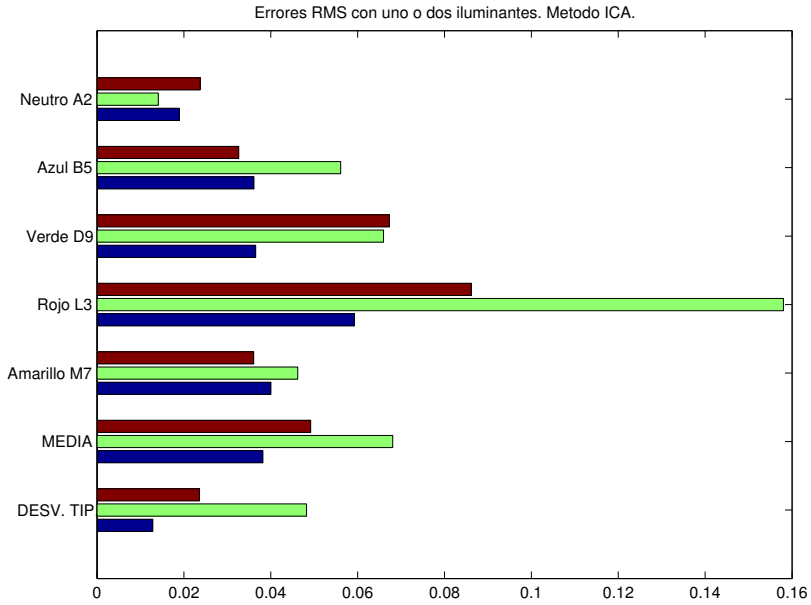

Figura 3.18: Errores *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método del análisis de componentes independientes.

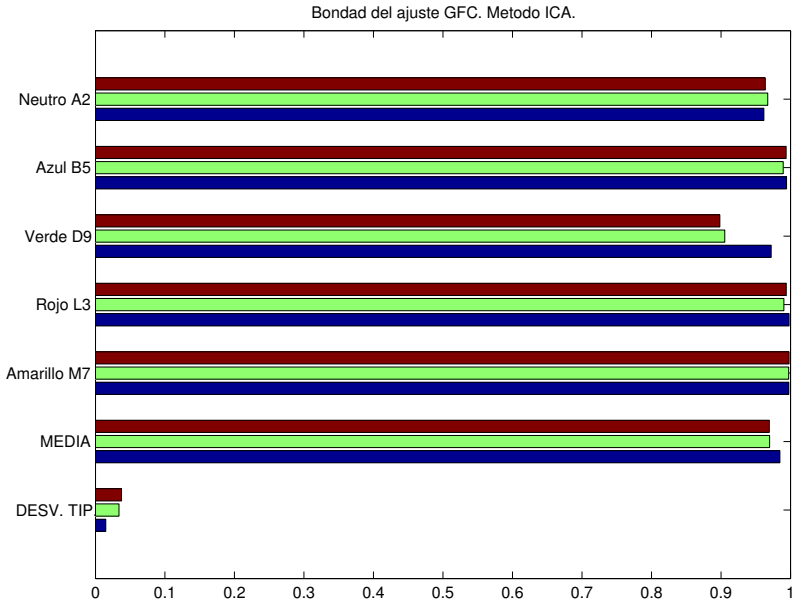

Figura 3.19: Coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método del análisis de componentes independientes.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.36	5.47	6.80
Azul B5	3.43	2.62	1.20
Verde D9	6.77	6.25	3.50
Rojo L3	3.97	10.69	1.73
Amarillo M7	1.16	4.07	2.01
MEDIA	4.73	5.82	3.04
DESV. TIP.	2.54	2.73	2.02

Tabla 3.12: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método del análisis de componentes independientes. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

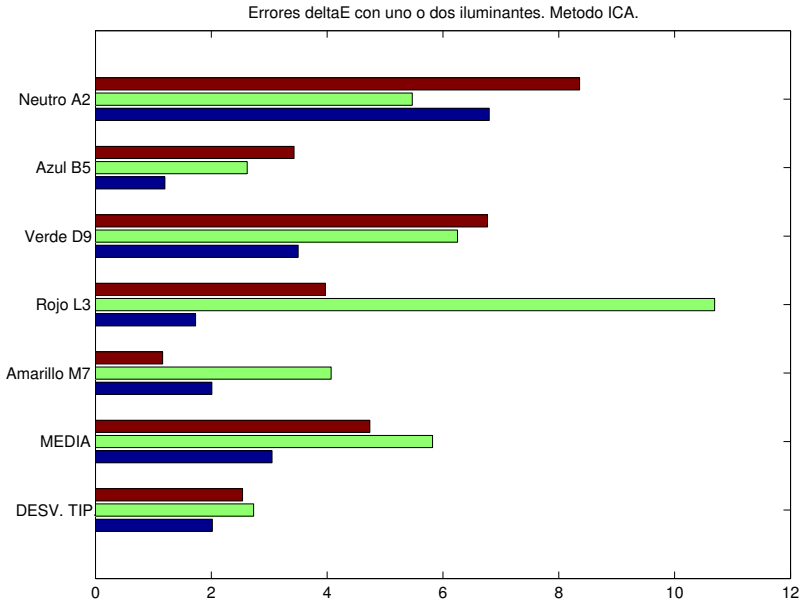

Figura 3.20: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método del análisis de componentes independientes. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	9.59	6.35	7.93
Azul B5	5.34	4.65	1.82
Verde D9	3.62	2.35	1.61
Rojo L3	0.66	7.72	0.79
Amarillo M7	0.85	3.20	1.48
MEDIA	4.01	4.85	2.73
DESV. TIP.	3.29	1.97	2.62

Tabla 3.13: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método del análisis de componentes independientes. El iluminante tomado como referencia para el cálculo del error perceptivo es el A (2856K).

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.41	5.64	6.88
Azul B5	2.98	2.42	1.22
Verde D9	8.58	8.78	4.72
Rojo L3	5.54	11.08	2.27
Amarillo M7	1.34	4.70	2.35
MEDIA	5.37	6.52	3.49
DESV. TIP.	2.88	3.06	2.05

Tabla 3.14: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método del análisis de componentes independientes. El iluminante tomado como referencia para el cálculo del error perceptivo es el D65.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	1.18	0.71	1.05
Azul B5	2.36	2.23	0.60
Verde D9	4.96	6.43	3.11
Rojo L3	4.88	3.36	1.48
Amarillo M7	0.49	1.50	0.87
MEDIA	2.77	2.85	1.42
DESV. TIP.	1.85	1.99	0.89

Tabla 3.15: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método del análisis de componentes independientes.

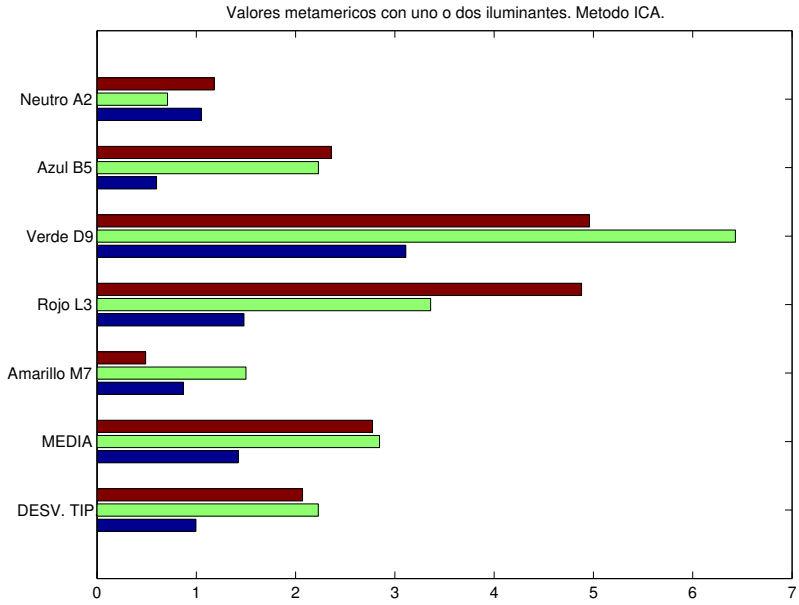

Figura 3.21: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método del análisis de componentes independientes.

3.4. Método de los colores cercanos (CC)

Siguiendo el procedimiento descrito en la sección 2.5 se ha determinado que, en las condiciones en las que se realiza este experimento, el número óptimo de parches de referencia para realizar la reconstrucción es de 44.

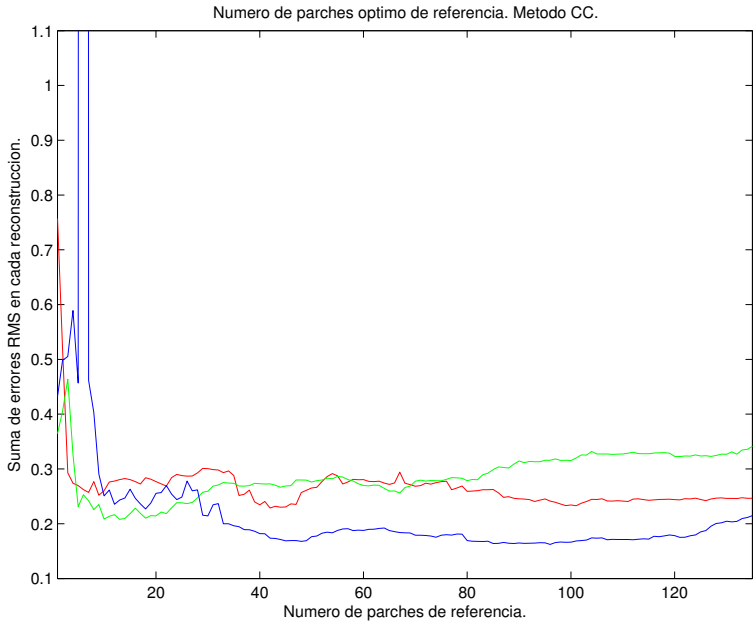

Figura 3.22: Suma de los errores *RMS* cometidos en la reconstrucción de los 5 paches de test a distintos números de parches de referencia con 1 único iluminante a 2700K (línea roja), 1 único iluminante a 6500K (línea verde) y 2 iluminantes (línea azul) por el método de los colores cercanos.

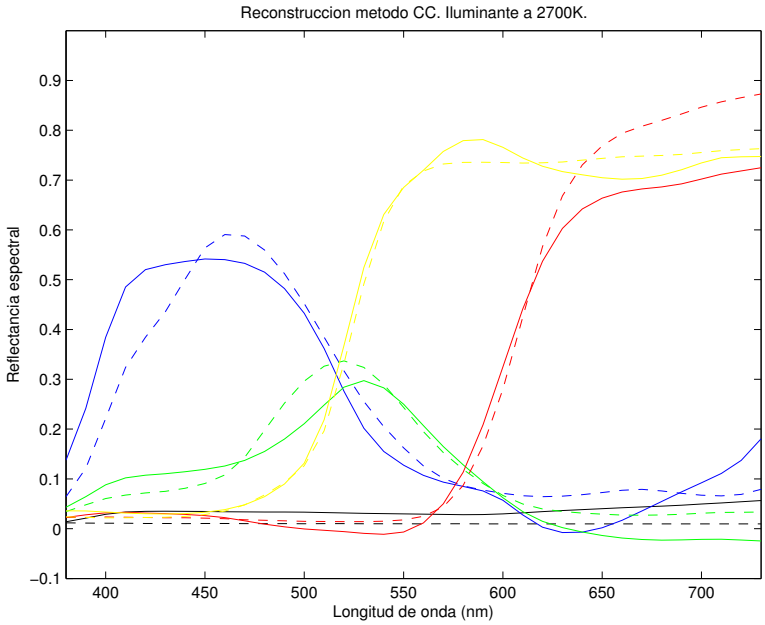

Figura 3.23: Reconstrucción de los 5 parches de test realizada con el método de los colores cercanos empleando 1 único iluminante a 2700K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

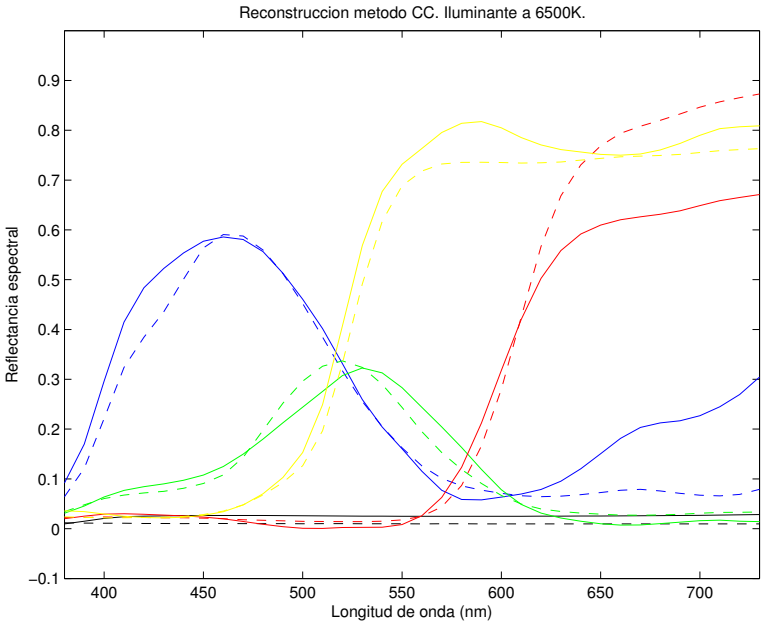

Figura 3.24: Reconstrucción de los 5 parches de test realizada con el método de los colores cercanos empleando 1 único iluminante a 6500K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

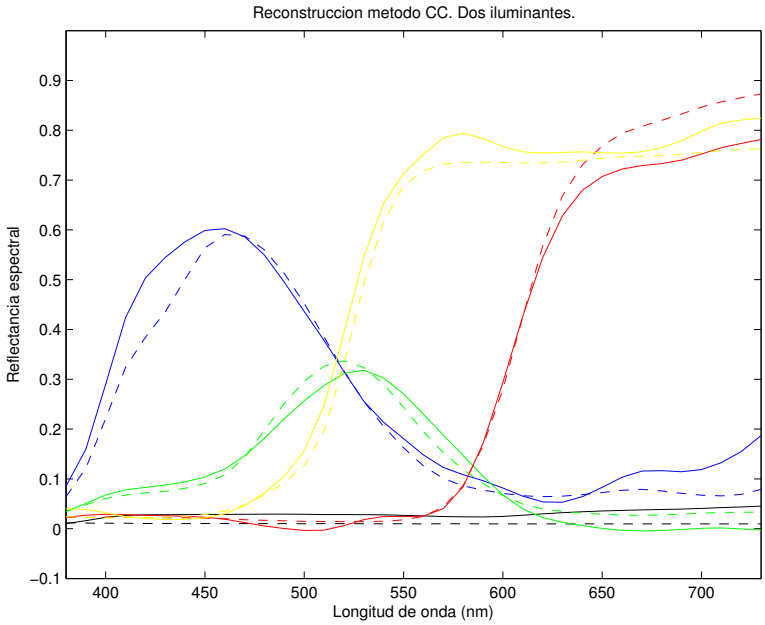

Figura 3.25: Reconstrucción de los 5 parches de test realizada con el método de los colores cercanos empleando 2 iluminantes distintos (6 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.0269/0.9625	0.0153/0.9845	0.0213/0.9624
Azul B5	0.0677/0.9744	0.0857/0.9682	0.0486/0.9906
Verde D9	0.0403/0.9658	0.0252/0.9863	0.0239/0.9875
Rojo L3	0.0717/0.9975	0.1005/0.9965	0.0446/0.9994
Amarillo M7	0.0235/0.9991	0.0400/0.9992	0.0328/0.9995
MEDIA	0.0460/0.9798	0.0533/0.9869	0.0324/0.9879
DESV. TIP.	0.0202/0.0156	0.0337/0.0110	0.0109/0.0136

Tabla 3.16: Errores *RMS* y coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de los colores cercanos.

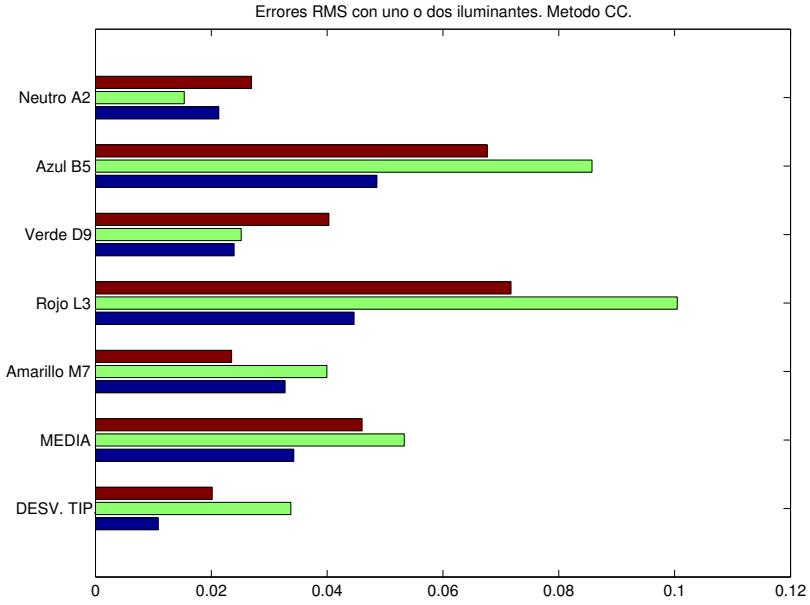

Figura 3.26: Errores *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de los colores cercanos.

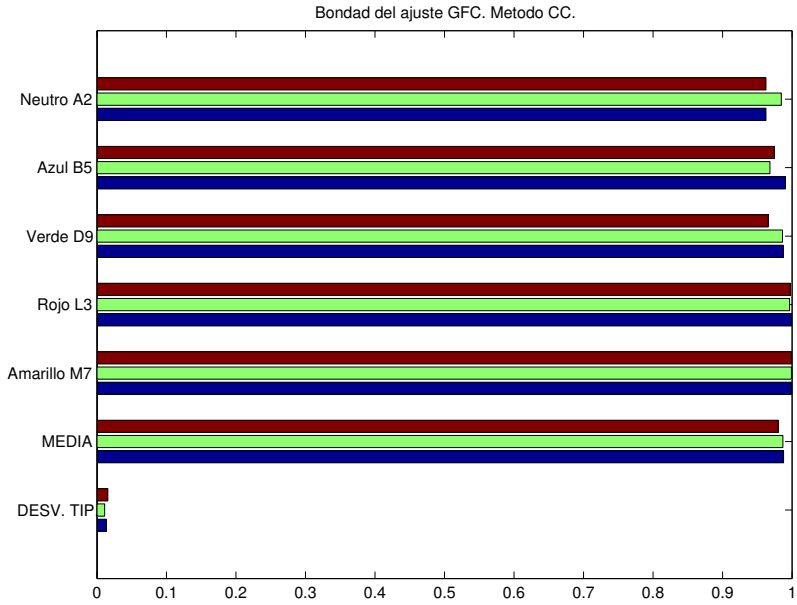

Figura 3.27: Coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de los colores cercanos.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.25	5.91	6.65
Azul B5	4.11	2.69	2.17
Verde D9	2.87	2.20	0.65
Rojo L3	2.40	1.16	0.94
Amarillo M7	0.99	2.54	2.09
MEDIA	3.72	2.90	2.50
DESV. TIP.	2.47	1.60	2.16

Tabla 3.17: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

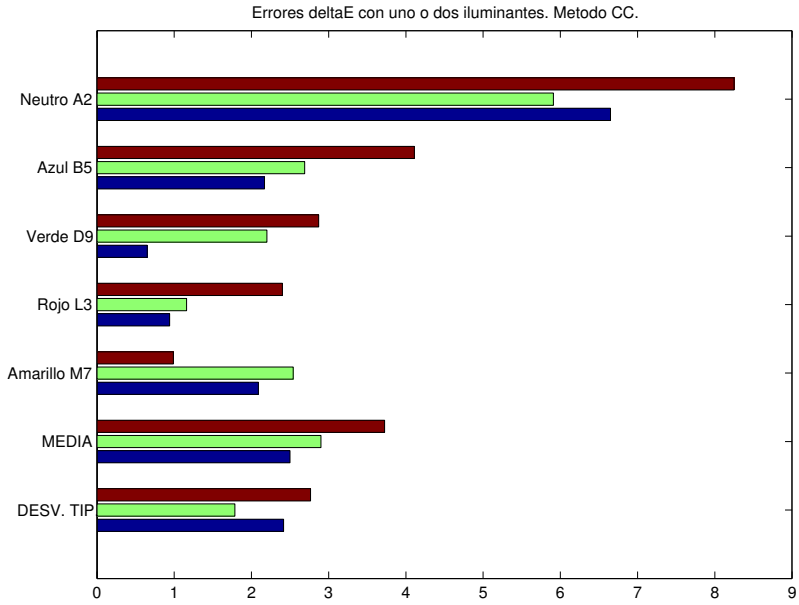

Figura 3.28: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	9.41	7.02	8.03
Azul B5	6.50	5.84	1.88
Verde D9	2.53	2.46	1.16
Rojo L3	1.11	1.89	1.15
Amarillo M7	0.93	2.23	1.71
MEDIA	4.10	3.89	2.79
DESV. TIP.	3.33	2.11	2.64

Tabla 3.18: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el A (2856K).

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	8.38	6.04	6.72
Azul B5	3.92	2.33	2.35
Verde D9	3.75	2.10	0.76
Rojo L3	3.02	0.94	0.75
Amarillo M7	1.00	2.61	2.81
MEDIA	4.01	2.80	2.68
DESV. TIP.	2.42	1.71	2.18

Tabla 3.19: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el D65.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	1.03	0.98	1.31
Azul B5	2.58	3.51	0.47
Verde D9	1.22	0.36	0.40
Rojo L3	1.91	0.95	0.40
Amarillo M7	0.07	0.38	1.10
MEDIA	1.36	1.24	0.73
DESV. TIP.	0.85	1.17	0.39

Tabla 3.20: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método de los colores cercanos.

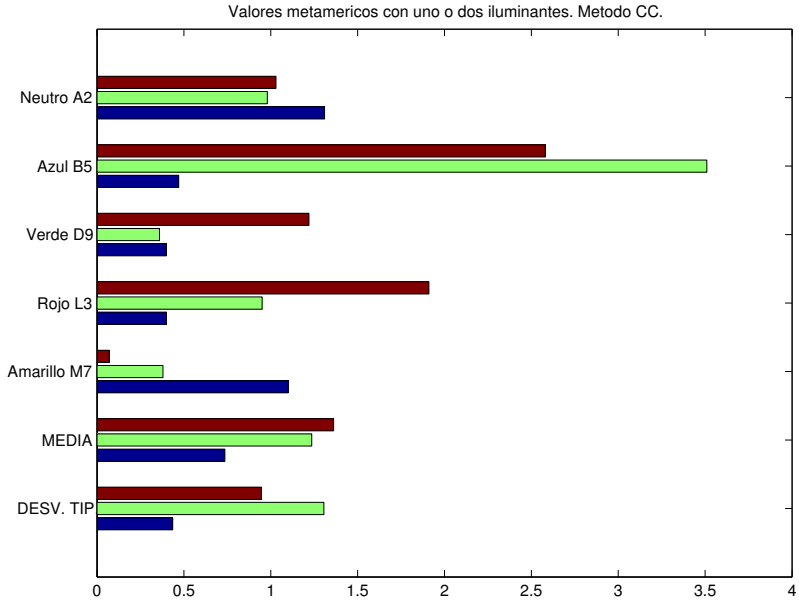

Figura 3.29: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método de los colores cercanos.

3.5. Método cuadrático de los colores cercanos (CCC)

Siguiendo el procedimiento descrito en la sección 2.6 se ha determinado que, en las condiciones en las que se realiza este experimento, el número óptimo de parches de referencia para realizar la reconstrucción es de 82.

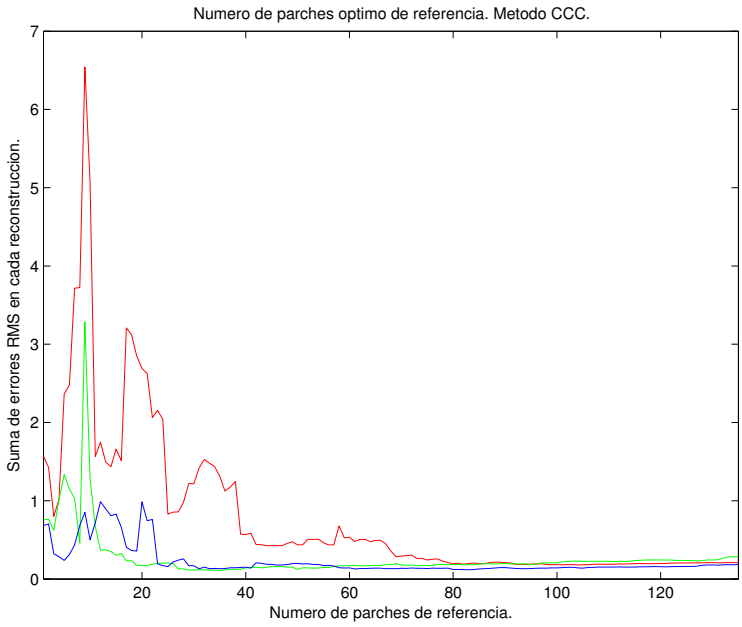

Figura 3.30: Suma de los errores *RMS* cometidos en la reconstrucción de los 5 paches de test a distintos números de parches de referencia con 1 único iluminante a 2700K (línea roja), 1 único iluminante a 6500K (línea verde) y 2 iluminantes (línea azul) por el método cuadrático de los colores cercanos.

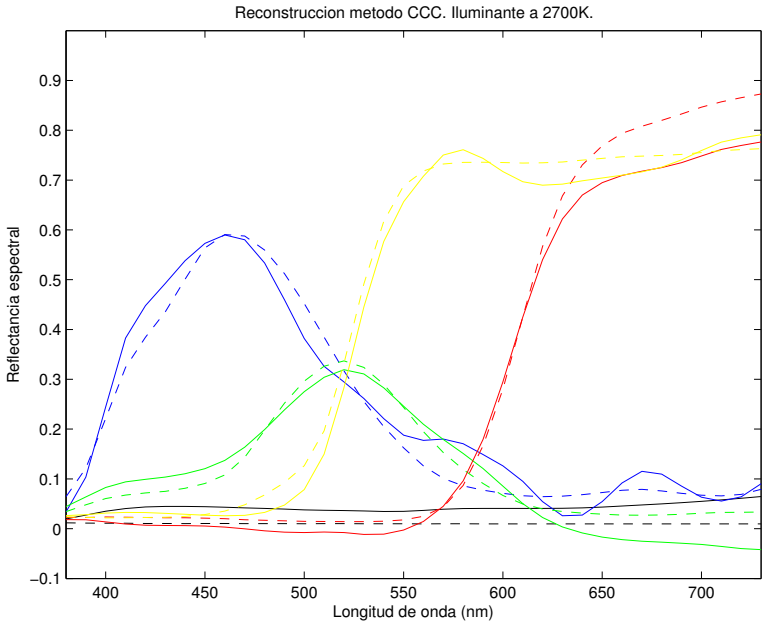

Figura 3.31: Reconstrucción de los 5 parches de test realizada con el método cuadrático de los colores cercanos empleando 1 único iluminante a 2700K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

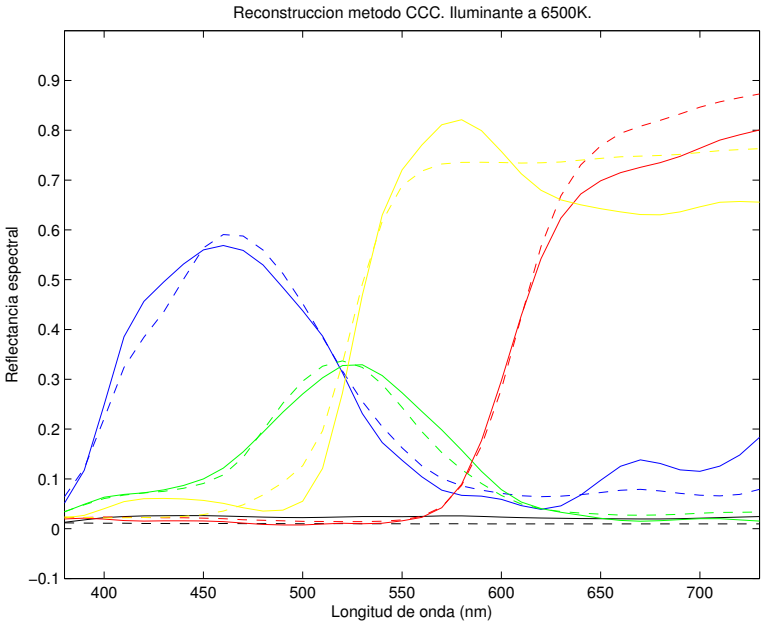

Figura 3.32: Reconstrucción de los 5 parches de test realizada con el método cuadrático de los colores cercanos empleando 1 único iluminante a 6500K (3 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas continuas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

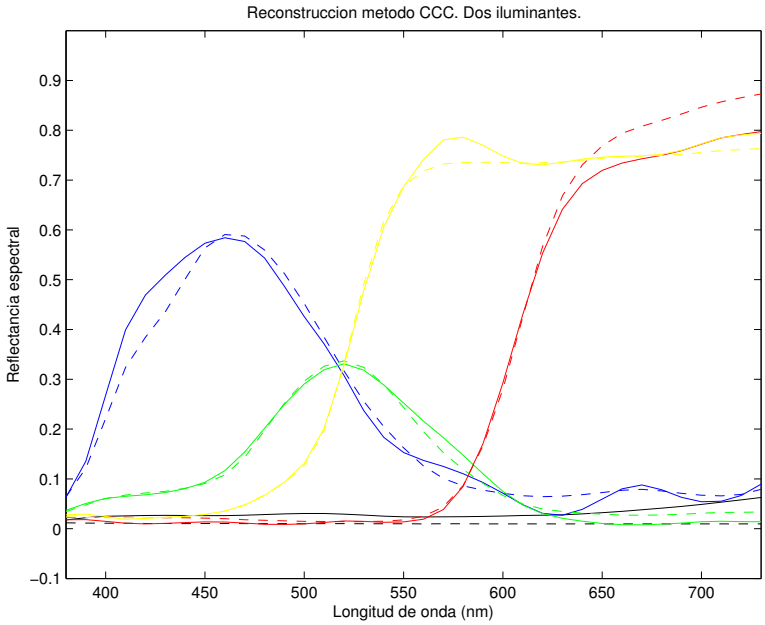

Figura 3.33: Reconstrucción de los 5 parches de test realizada con el método cuadrático de los colores cercanos empleando 2 iluminantes distintos (6 canales). Las líneas a trazos representan las curvas de reflectancia originales de los parches mientras que las líneas reconstruidas representan las curvas de reflectancia reconstruidas. Sirva para indicar cualitativamente y de forma aproximada el color del parche sobre el que se trabaja.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.0334/0.9730	0.0130/0.9903	0.0235/0.9416
Azul B5	0.0405/0.9906	0.0400/0.9909	0.0297/0.9952
Verde D9	0.0362/0.9720	0.0173/0.9940	0.0137/0.9960
Rojo L3	0.0499/0.9990	0.0418/0.9996	0.0358/0.9996
Amarillo M7	0.0292/0.9989	0.0694/0.9936	0.0167/0.9997
MEDIA	0.0378/0.9867	0.0363/0.9937	0.0239/0.9864
DESV. TIP.	0.0071/0.0120	0.0202/0.0033	0.0081/0.0225

Tabla 3.21: Errores *RMS* y coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método cuadrático de los colores cercanos.

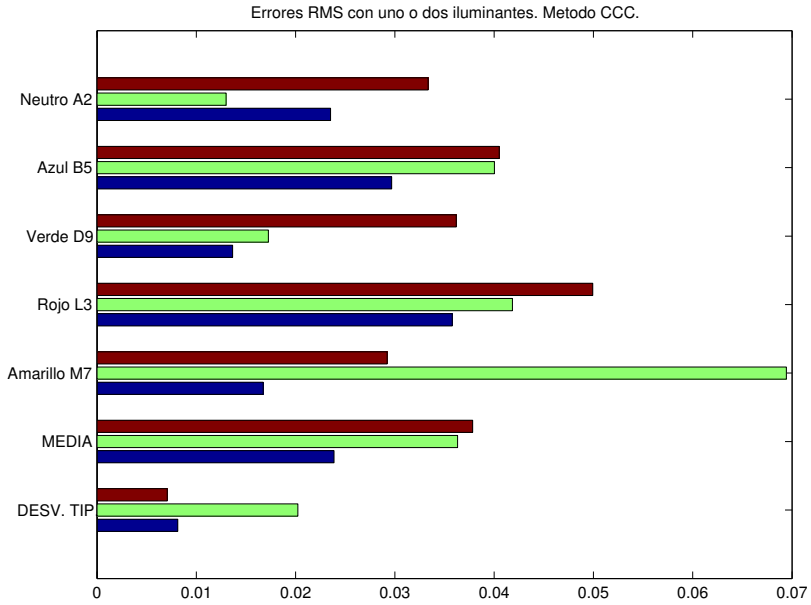

Figura 3.34: Errores *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método cuadrático de los colores cercanos.

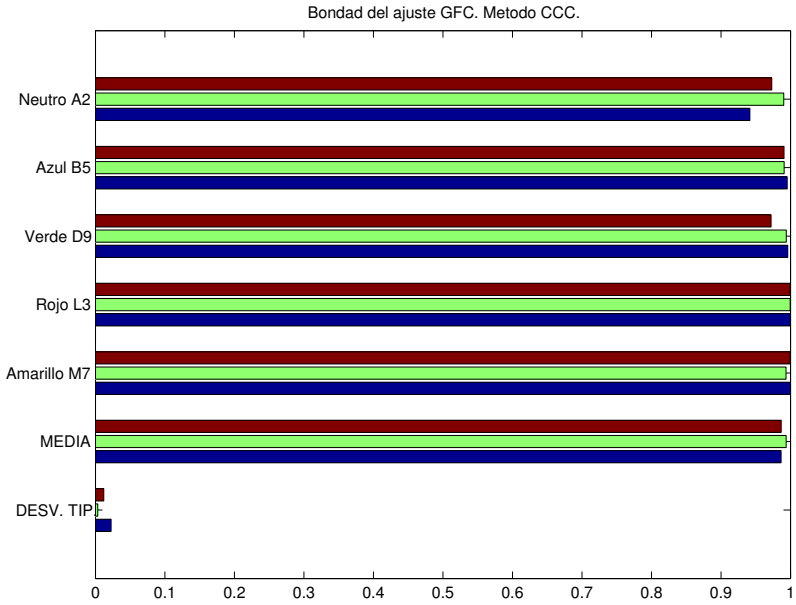

Figura 3.35: Coeficientes *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método cuadrático de los colores cercanos.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	10.60	5.59	6.24
Azul B5	4.16	2.74	1.20
Verde D9	2.30	2.38	0.92
Rojo L3	7.28	2.65	3.42
Amarillo M7	1.32	0.67	0.48
MEDIA	5.13	2.80	2.45
DESV. TIP.	3.41	1.58	2.15

Tabla 3.22: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método cuadrático de los colores cercanos.El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

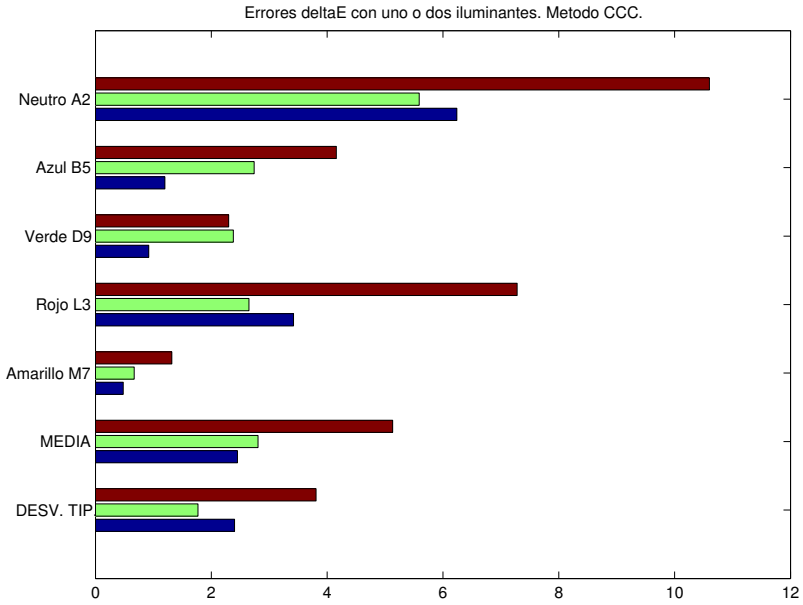

Figura 3.36: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método cuadrático de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el D50.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	11.49	6.50	7.66
Azul B5	5.56	3.42	1.27
Verde D9	1.38	2.76	1.01
Rojo L3	3.40	1.74	1.84
Amarillo M7	0.91	1.34	0.50
MEDIA	4.55	3.15	2.45
DESV. TIP.	3.85	1.83	2.63

Tabla 3.23: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método cuadrático de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el A (2856K).

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	10.83	5.73	6.32
Azul B5	4.14	2.68	1.50
Verde D9	3.16	2.20	0.90
Rojo L3	8.96	3.00	4.08
Amarillo M7	1.59	1.14	0.51
MEDIA	5.74	2.95	2.66
DESV. TIP.	3.54	1.52	2.21

Tabla 3.24: Errores $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método cuadrático de los colores cercanos. El iluminante tomado como referencia para el cálculo del error perceptivo es el D65.

Parche	1 Ilum (2700K)	1 Ilum (6500K)	2 Ilum.
Neutro A2	0.66	0.77	1.34
Azul B5	1.42	0.74	0.23
Verde D9	1.78	0.56	0.11
Rojo L3	5.56	1.26	2.24
Amarillo M7	0.68	0.20	0.01
MEDIA	2.02	0.71	0.78
DESV. TIP.	1.82	0.34	0.87

Tabla 3.25: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K, un iluminante a 6500K y dos iluminantes por el método cuadrático de los colores cercanos.

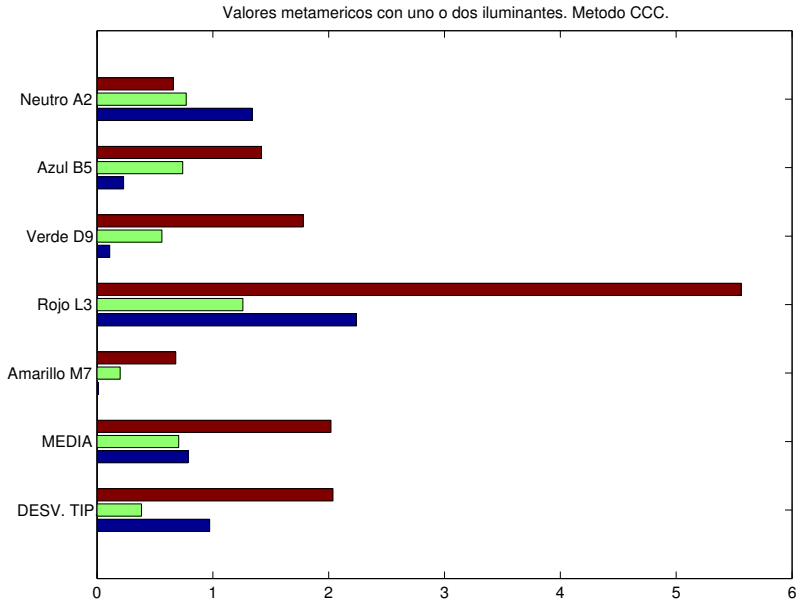

Figura 3.37: Valores metaméricos obtenidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por el método cuadrático de los colores cercanos.

3.6. Resumen de los resultados

Metodo	Error	1 Ilum. 2700K	1 Ilum. 6500K	2 Ilum.
PINV	RMS	0.0492	0.0682	0.0429
	GFC	0.9709	0.9694	0.9835
	ΔE	5.11	4.23	3.41
	VM	2.88	2.10	1.40
SVD	RMS	0.0493	0.0682	0.0430
	GFC	0.9709	0.9693	0.9837
	ΔE	5.11	4.23	3.46
	VM	2.88	2.11	1.49
ICA	RMS	0.0492	0.0681	0.0382
	GFC	0.9696	0.9700	0.9848
	ΔE	4.73	5.82	3.04
	VM	2.77	2.85	1.42
CC	RMS	0.0460	0.0533	0.0324
	GFC	0.9798	0.9869	0.9879
	ΔE	3.72	2.90	2.50
	VM	1.36	1.24	0.73
CCC	RMS	0.0378	0.0363	0.0239
	GFC	0.9867	0.9937	0.9864
	ΔE	5.13	2.80	2.45
	VM	2.02	0.71	0.78

Tabla 3.26: Tabla resumen de los errores medios *DeltaE* y *RMS*, de los coeficientes de bondad de ajuste *GFC* y de los valores metaméricos medios correspondientes a las reconstrucciones con los diferentes métodos con 1 iluminante a 2700K, con 1 iluminante a 6500K y con 2 iluminantes.

Metodo	D.T.	1 Ilum. 2700K	1 Ilum. 6500K	2 Ilum.
PINV	*RMS*	0.0248	0.0485	0.0163
	GFC	0.0339	0.0314	0.0159
	ΔE	2.72	1.24	1.80
	VM	2.19	1.27	1.04
SVD	*RMS*	0.0248	0.0486	0.0162
	GFC	0.0339	0.0314	0.0157
	ΔE	2.71	1.25	1.78
	VM	2.19	1.28	0.99
ICA	*RMS*	0.0236	0.0482	0.0129
	GFC	0.0376	0.0339	0.0149
	ΔE	2.54	2.73	2.02
	VM	1.85	1.99	0.89
CC	*RMS*	0.0202	0.0337	0.0109
	GFC	0.0156	0.0110	0.0136
	ΔE	2.47	1.60	2.16
	VM	0.85	1.17	0.39
CCC	*RMS*	0.0071	0.0202	0.0081
	GFC	0.0120	0.0033	0.0225
	ΔE	3.41	1.58	2.15
	VM	1.82	0.34	0.87

Tabla 3.27: Tabla resumen de las desviaciones típicas los errores *DeltaE* y *RMS*, de las desviaciones típicas de los coeficientes de bondad de ajuste *GFC* y de las desviaciones típicas de los valores metaméricos correspondientes a las reconstrucciones con los diferentes métodos con 1 iluminante a 2700K, con 1 iluminante a 6500K y con 2 iluminantes.

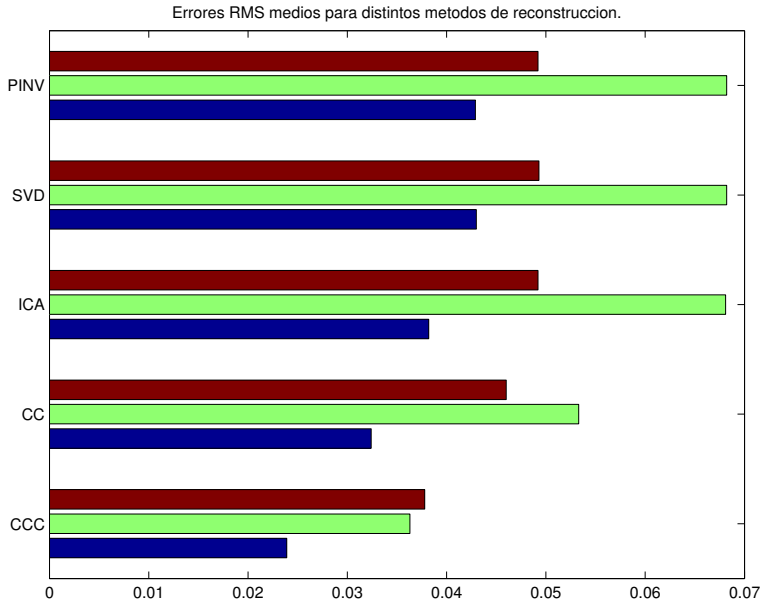

Figura 3.38: Errores medios *RMS* cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

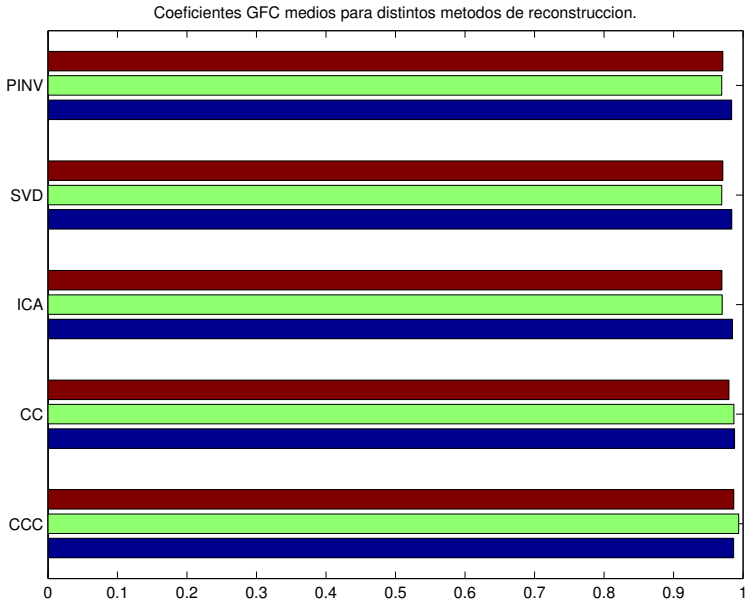

Figura 3.39: Coeficientes de bondad de ajuste medios *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

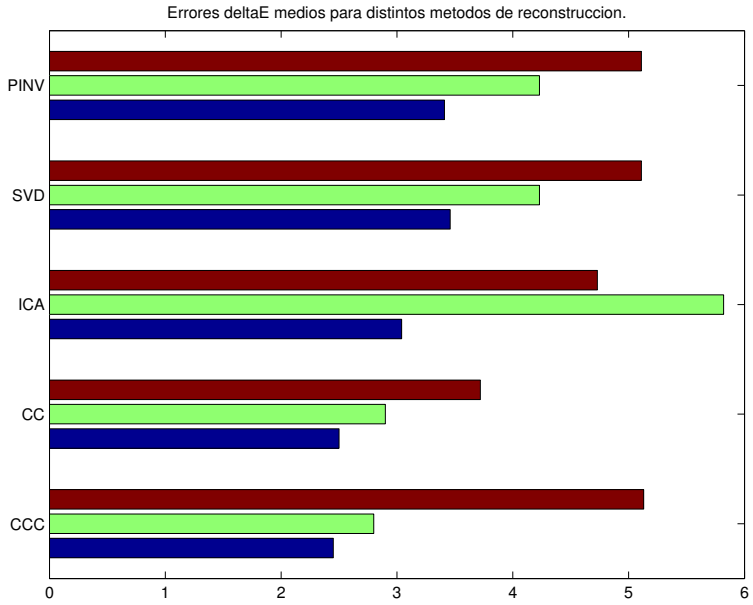

Figura 3.40: Errores medios $\Delta E2000$ cometidos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

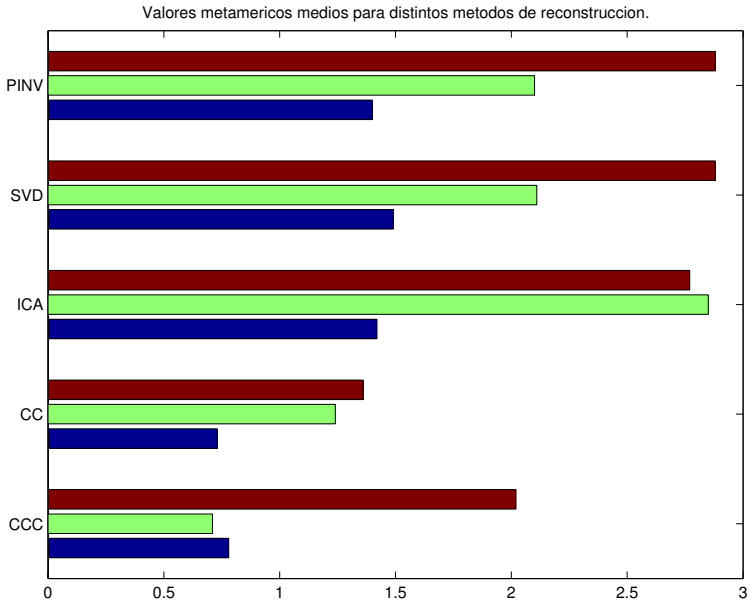

Figura 3.41: Valores metaméricos medios de la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

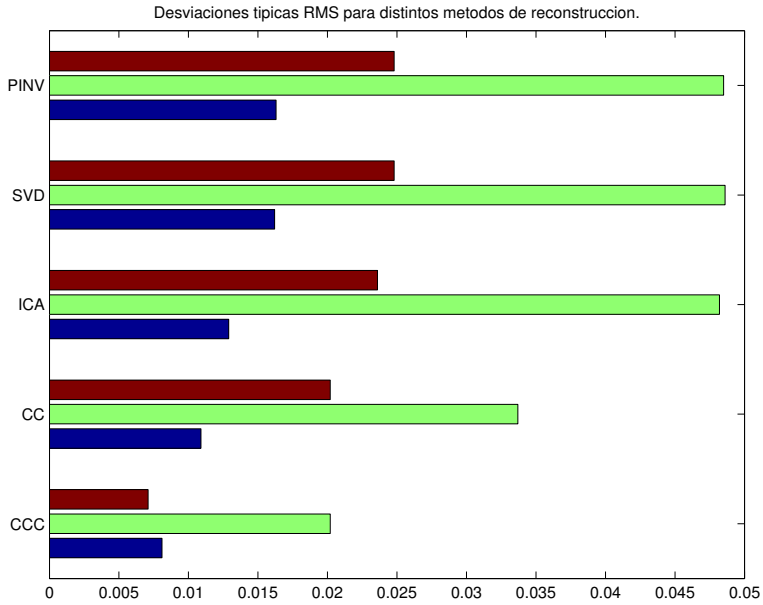

Figura 3.42: Desviaciones típicas medias *RMS* cometidas en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

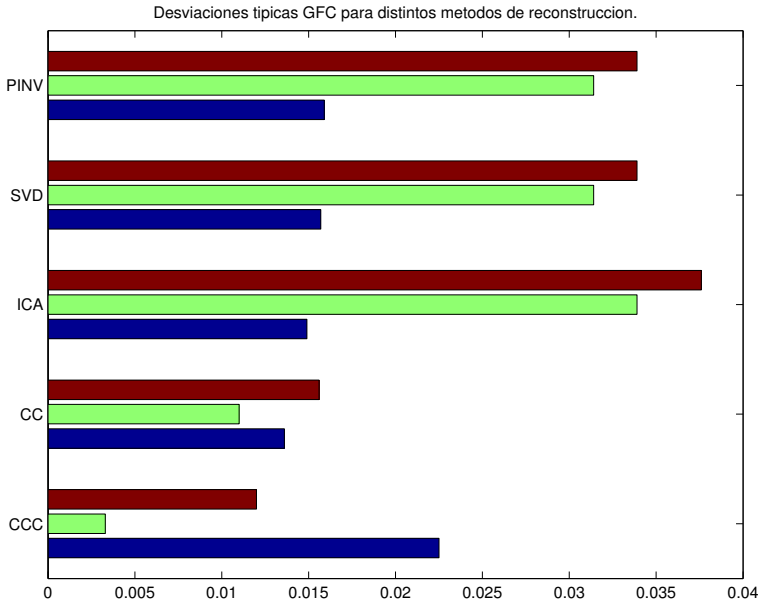

Figura 3.43: Desviaciones típicas de los coeficientes de bondad de ajuste *GFC* en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

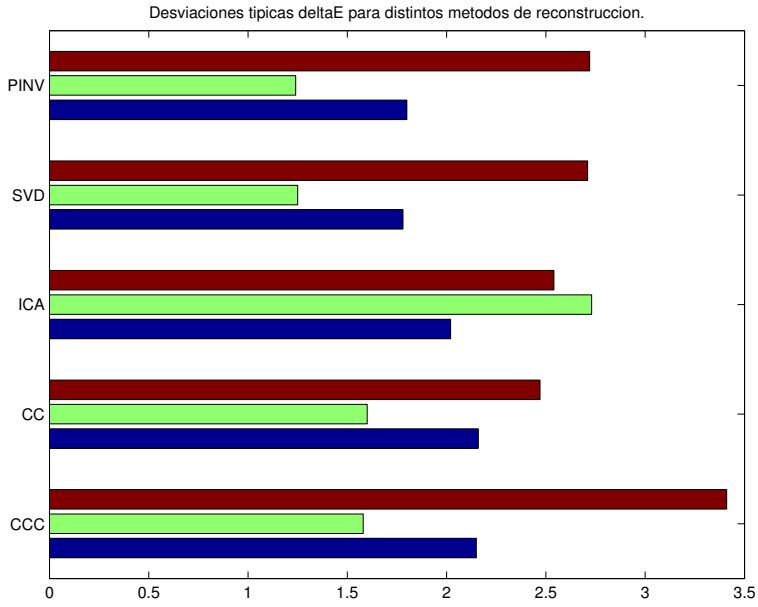

Figura 3.44: Desviaciones típicas medias $\Delta E2000$ cometidas en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

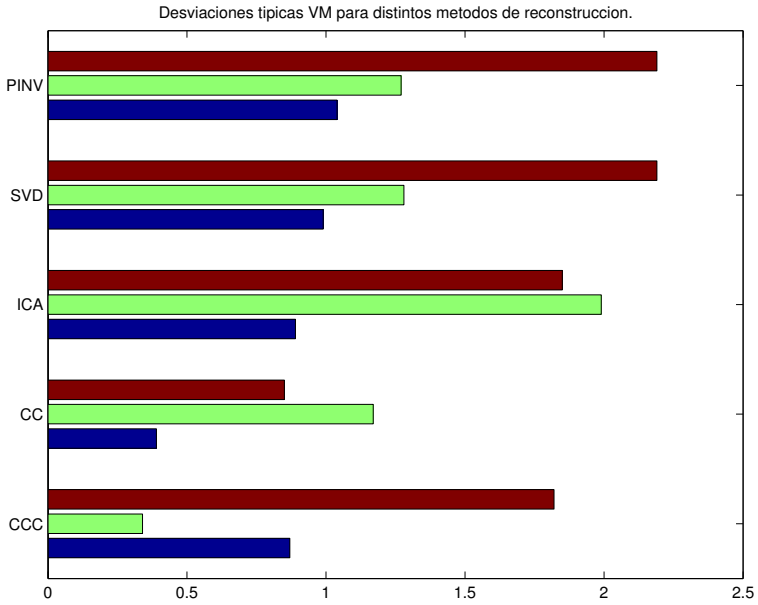

Figura 3.45: Desviaciones típicas medias de los valores metaméricos en la reconstrucción de los parches de test con un iluminante a 2700K (barras en rojo), un iluminante a 6500K (barras en verde) y dos iluminantes (barras en azul) por los distintos métodos.

Capítulo 4

Discusión

La medida de la reflectancia espectral de objetos a partir de una cámara digital calibrada es un objetivo industrial y tecnológico desde hace 20 años, pero que sigue vigente en la actualidad puesto que la solución definitiva, si bien se ha progresado muchísimo, no se ha alcanzado todavía al 100 %. Los avances actuales en este campo se han centrado en la tecnología hyper o multi-spectral imaging que consiste básicamente en aumentar el nº de canales en la captura de imagen, de 3 a 6, o mucho más, como 15 ó 20.

El objetivo de esta tesis se centra en la aportación innovadora de aumentar el nº de canales a partir del aumento de toma de imágenes (canales), pero no a nivel espectral, sino usando la variabilidad espectral de la fuente de luz, o sea variando las condiciones de iluminación, y no las condiciones de captura o diseño óptico-espectral de la cámara digital.

El uso de dos iluminantes en lugar de uno solo permite reducir el error espectral entre un 12 % y un 30 % y el perceptivo entre un 13 % y un 35 % en el peor de los casos[1]. La bondad del ajuste gana entre 2 y 3 centésimas utilizando dos iluminantes o empleando el método CC propuesto en el presente trabajo. Además, con dicho método y con su variante cuadrática se disminuye en más de un 28 %[2] el error perceptivo

[1] Es decir, comparando el error cometido con dos iluminantes con el error más bajo que se puede cometer con uno solo.

[2] De nuevo consideramos el peor de los casos que se da con dos iluminantes.

cometido en la reconstrucción de los parches de test respecto al método de la pseudoinversa que cabe tomar como referencia por tratarse del más obvio de todos. Desde el punto de vista del error espectral, los resultados obtenidos con el método cuadrático de los colores cercanos son un 32 % más precisos en el peor de los casos. El único problema que plantea el método CC y el método CCC sería el de su complejidad, pero este pequeño inconveniente –se intuye salvable gracias al estado de la tecnología actual– tiene como recompensa una reconstrucción espectral mucho más precisa.

Además, la reconstrucción presenta mejor comportamiento con una reducción del metamerismo de hasta el 50 % con los métodos propuestos en estre trabajo y el 47 % con dos iluminantes en lugar de uno solo.

El valor metamérico obtenido en la reconstrucción a 6500K con un iluminante y método ICA, es superior (más elevado) al obtenido con PINV en las mismas condiciones, lo que indica que –al menos– para reconstrucciones espectrales con tres canales, la carta de color ColorChecker SG empleada cumple su cometido. Para la reconstrucción a 2700K vemos que la diferencia cambia pero esto podría deberse a las condiciones comparativamente peores en las que se realiza el revelado de la fotografía de donde se obtienen las respuestas tricromáticas, ya que el revelado en este caso se hace a una temperatura que no se corresponde con la temperatura de las lámparas. Por otro lado, para reconstrucciones con seis canales vemos que la diferencia se pierde, lo que podría ser atribuído al grado de explicación elevado que tienen por si mismas las recontrucciones con seis canales respecto a las curvas espectrales originales de los parches.

El método PINV debería emplearse para obtener una primera aproximación a la respuesta. El método ICA puede emplearse cuando es posible presuponer que existen determinados colores base que mezclados entre si producen aquellos que se desea reconstruir, pero no en el resto de casos, ya que se trata de un método sensible al metamerismo y computacionalmente costoso. Para reconstrucciones precisas con un bajo valor metamérico se recomienda el uso del método CC o su variante cuadrática CCC expuestos

en el presente trabajo de investigación.

La medición del color con cámara fotográfica doméstica tiene diversas ventajas con respecto a los espectrorradiómetros: su precio es más bajo, es más robusta y fácil de usar y proporciona medidas sobre una área más extensa. Las técnicas propuestas y ensayadas ofrecen unas prestaciones que mejoran el estado de la tecnología actual y que podrían ser objeto de inclusión en dichos dispositivos. De cara a futuros trabajos podría tratar de implementarse un sistema basado en el método de los colores cercanos o en cualquiera de sus versiones no lineales en dispositivos móviles (tabletas o teléfonos) o en escáneres con uno o con dos iluminantes. Podría contruirse un prototipo basado en el escáner empleado por [1].

No se ha explicado por qué la cámara usa un espacio RGB identico al sRGB estandarizado. El presente trabajo presupone que el espacio sRGB que da la cámara coincide con el estandarizado y partiendo de esa suposición se realiza un análisis de resultados sobre la reconstrucción espectral de un conjunto de parches por diferentes métodos, con el objeto no de obtener resultados excentes en términos absolutos, sino de comparar la bondad de los resultados entre los distintos métodos. La cámara empleada es una cámara de uso doméstico y buena parte de su funcionamiento interno es desconocido para los autores.

Se propone como posible trabajo futuro de investigación discutir sobre la influencia de la temperatura correlacionada de color (Tc) y el rendimiento en color (Ra) en los resultados.

El equipo que ha realizado el presente trabajo está preparando unos artículos científicos con la intención de publicarlos en alguna revista de investigación del ámbito del color.

Se puede concluir pues, que el objetivo de reconstruir las curvas de reflectancia espectral de un conjunto de parches de test empleando dos iluminantes y una cámara doméstica ha sido logrado con éxito obteniendo valores de error similares o mejores a los obtenidos por otros autores en la literatura. Además, los métodos CC y CCC propuestos en el presente trabajo ofrecen mejores resultados que los métodos PINV, SVD e ICA.

Capítulo 5

Anexos

5.1. Código del método PINV

```
% Reconstruccion metodo pseudoinversa iluminante 2700K

X3ch2700=pinv(XYZtraining2700)*SPKTtraining;
reconstructed3ch2700=XYZcontrol2700*X3ch2700;
figure;
plot([380:10:730], (reconstructed3ch2700(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch2700(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch2700(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch2700(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch2700(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo pinv. Iluminante a 2700K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo pseudoinversa iluminante 6500K

X3ch6500=pinv(XYZtraining6500)*SPKTtraining;
reconstructed3ch6500=XYZcontrol6500*X3ch6500;
figure;
plot([380:10:730], (reconstructed3ch6500(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch6500(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
```

```
    [380:10:730], (reconstructed3ch6500(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch6500(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch6500(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo pinv. Iluminante a 6500K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo PINV dos iluminantes (2700K y 6500K)

X6ch=pinv(XYZtraining)*SPKTtraining;
reconstructed6ch=XYZcontrol*X6ch;
figure;
plot([380:10:730], (reconstructed6ch(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed6ch(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed6ch(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed6ch(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed6ch(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo pinv. Dos iluminantes.');
axis([380, 730, -0.1, 1]);

% Calculo del error RMS

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
```

```
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];

% Calculo de la media y la desviacion tipica

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(suma / 5);
RMS=[RMS; RMSmean; RMSdesv];

% Graficar errores RMS

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(RMS)), 'group');
title('Errores RMS con uno o dos iluminantes. Metodo PINV.');
set(gca, 'YTickLabel', colores);

% Calcula los GFC

Num = sum((SPKTcontrol.*reconstructed3ch2700), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch2700.*reconstructed3ch2700), 2));
gfc3ch2700 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed3ch6500), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch6500.*reconstructed3ch6500), 2));
gfc3ch6500 = Num ./ Den;
```

```
Num = sum((SPKTcontrol.*reconstructed6ch), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed6ch.*reconstructed6ch), 2));
gfc6ch = Num ./ Den;

gfc_full=[gfc3ch2700, gfc3ch6500, gfc6ch];

GFCmean=mean(gfc_full);
suma = 0;
for(i=1:5)
    suma = suma + (gfc_full(i, :) - GFCmean) .^ 2;
end
GFCdesv = sqrt(suma / 5);

gfc_full=[gfc_full; GFCmean; GFCdesv]

% Graficar gfc

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(gfc_full)), 'group');
title('Bondad del ajuste GFC. Metodo PINV.');
set(gca, 'YTickLabel', colores);

clear gfc3ch2700 gfc3ch6500 gfc6ch;
clear Num Den GFCmean GFCdesv i suma;
```

Después de introducir estas instrucciones en MATLAB conviene trasladar las curvas de reflectancia espectral reconstruidas a ficheros .txt de ColorLab como se indica en la sección 2.1.11 para realizar el cálculo de los errores $\Delta E2000$. Una vez realizado esto, se calculan los errores $\Delta E2000$ como se indica dentro de 2.1.7 y después se trasladan los resultados del

informe .txt generado por ColorLab a una matriz de MATLAB (deltaE) siguiendo las indicaciones de la sección 2.1.10. La matriz deltaE será la entrada que se necesita para ejecutar las siguientes instrucciones de MATLAB orientadas a graficar los errores $\Delta E2000$ cometidos en la reconstrucción.

```
% Calculo de la media y la desviacion tipica

deltaEmean=mean(deltaE);
suma = 0;
for(i=1:5)
    suma = suma + (deltaE(i, :) - deltaEmean) .^ 2;
end
deltaEdesv = sqrt(suma / 5);
deltaE=[deltaE; deltaEmean; deltaEdesv];

% Graficar errores deltaE

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(deltaE)), 'group');
title('Errores deltaE con uno o dos iluminantes. Metodo PINV.');
set(gca, 'YTickLabel', colores);
```

5.2. Código del método SVD

```
% Reconstruccion metodo SVD iluminante 2700K
```

```
[U, S, V]=svds(SPKTtraining, 6);
X3ch2700=pinv(XYZtraining2700)*U;
reconstructed3ch2700=XYZcontrol2700*X3ch2700*S*V';
figure;
plot([380:10:730], (reconstructed3ch2700(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch2700(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch2700(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch2700(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch2700(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo SVD. Iluminante a 2700K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo SVD iluminante 6500K

[U, S, V]=svds(SPKTtraining, 6);
X3ch6500=pinv(XYZtraining6500)*U;
reconstructed3ch6500=XYZcontrol6500*X3ch6500*S*V';
figure;
plot([380:10:730], (reconstructed3ch6500(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch6500(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch6500(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch6500(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
```

```
    [380:10:730], (reconstructed3ch6500(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo SVD. Iluminante a 6500K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo SVD dos iluminantes (2700K y 6500K)

[U, S, V]=svds(SPKTtraining, 6);
X6ch=pinv(XYZtraining)*U;
reconstructed6ch=XYZcontrol*X6ch*S*V';
figure;
plot([380:10:730], (reconstructed6ch(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed6ch(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed6ch(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed6ch(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed6ch(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo SVD. Dos iluminantes.');
axis([380, 730, -0.1, 1]);

% Calculo del error RMS

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];
```

```
% Calculo de la media y la desviacion tipica

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(suma / 5);
RMS=[RMS; RMSmean; RMSdesv];

% Graficar errores RMS

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(RMS)), 'group');
title('Errores RMS con uno o dos iluminantes. Metodo SVD.');
set(gca, 'YTickLabel', colores);

% Calcula los GFC

Num = sum((SPKTcontrol.*reconstructed3ch2700), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch2700.*reconstructed3ch2700), 2));
gfc3ch2700 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed3ch6500), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch6500.*reconstructed3ch6500), 2));
gfc3ch6500 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed6ch), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
```

```
 sqrt(sum((reconstructed6ch.*reconstructed6ch), 2));
gfc6ch = Num ./ Den;

gfc_full=[gfc3ch2700, gfc3ch6500, gfc6ch];

GFCmean=mean(gfc_full);
suma = 0;
for(i=1:5)
    suma = suma + (gfc_full(i, :) - GFCmean) .^ 2;
end
GFCdesv = sqrt(suma / 5);

gfc_full=[gfc_full; GFCmean; GFCdesv]

% Graficar gfc

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(gfc_full)), 'group');
title('Bondad del ajuste GFC. Metodo SVD.');
set(gca, 'YTickLabel', colores);

clear gfc3ch2700 gfc3ch6500 gfc6ch;
clear Num Den GFCmean GFCdesv i suma;
```

Después de introducir estas instrucciones en MATLAB conviene trasladar las curvas de reflectancia espectral reconstruidas a ficheros .txt de ColorLab como se indica en la sección 2.1.11 para realizar el cálculo de los errores $\Delta E2000$. Una vez realizado esto, se calculan los errores $\Delta E2000$ como se indica dentro de 2.1.7 y después se trasladan los resultados del informe .txt generado por ColorLab a una matriz de MATLAB (deltaE) siguiendo las indicaciones de la sección 2.1.10. La matriz deltaE será la entrada que se necesita para ejecutar las siguientes instrucciones

de MATLAB orientadas a graficar los errores $\Delta E2000$ cometidos en la reconstrucción.

```
% Calculo de la media y la desviacion tipica

deltaEmean=mean(deltaE);
suma = 0;
for(i=1:5)
    suma = suma + (deltaE(i, :) - deltaEmean) .^ 2;
end
deltaEdesv = sqrt(suma / 5);
deltaE=[deltaE; deltaEmean; deltaEdesv];

% Graficar errores deltaE

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(deltaE)), 'group');
title('Errores deltaE con uno o dos iluminantes. Metodo SVD.');
set(gca, 'YTickLabel', colores);
```

5.3. Código del método ICA

```
% Reconstruccion metodo ICA iluminante 2700K

[icasig, mix, W] = fastica (SPKTtraining, 'lastEig', 3,...
    'maxNumIterations', 100000);
```

```
X3ch2700=pinv(XYZtraining2700)*mix;
reconstructed3ch2700=XYZcontrol2700*X3ch2700*icasig;
figure;
plot([380:10:730], (reconstructed3ch2700(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch2700(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch2700(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch2700(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch2700(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo ICA. Iluminante a 2700K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo ICA iluminante 6500K

[icasig, mix, W] = fastica (SPKTtraining, 'lastEig', 3,...
    'maxNumIterations', 100000);
X3ch6500=pinv(XYZtraining6500)*mix;
reconstructed3ch6500=XYZcontrol6500*X3ch6500*icasig;
figure;
plot([380:10:730], (reconstructed3ch6500(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch6500(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch6500(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch6500(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch6500(5, :))', 'y-',...
```

```
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo ICA. Iluminante a 6500K.');
axis([380, 730, -0.1, 1]);

% Reconstruccion metodo ICA dos iluminantes (2700K y 6500K)

[icasig, mix, W] = fastica (SPKTtraining, 'lastEig', 6,...
    'maxNumIterations', 100000);
X6ch=pinv(XYZtraining)*mix;
reconstructed6ch=XYZcontrol*X6ch*icasig;
figure;
plot([380:10:730], (reconstructed6ch(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed6ch(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed6ch(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed6ch(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed6ch(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo ICA. Dos iluminantes.');
axis([380, 730, -0.1, 1]);

% Calculo del error RMS

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];
```

```
% Calculo de la media y la desviacion tipica

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(suma / 5);
RMS=[RMS; RMSmean; RMSdesv];

% Graficar errores RMS

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(RMS)), 'group');
title('Errores RMS con uno o dos iluminantes. Metodo ICA.');
set(gca, 'YTickLabel', colores);


% Calcula los GFC

Num = sum((SPKTcontrol.*reconstructed3ch2700), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch2700.*reconstructed3ch2700), 2));
gfc3ch2700 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed3ch6500), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch6500.*reconstructed3ch6500), 2));
gfc3ch6500 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed6ch), 2);
```

```
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed6ch.*reconstructed6ch), 2));
gfc6ch = Num ./ Den;

gfc_full=[gfc3ch2700, gfc3ch6500, gfc6ch];

GFCmean=mean(gfc_full);
suma = 0;
for(i=1:5)
    suma = suma + (gfc_full(i, :) - GFCmean) .^ 2;
end
GFCdesv = sqrt(suma / 5);

gfc_full=[gfc_full; GFCmean; GFCdesv]

% Graficar gfc

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(gfc_full)), 'group');
title('Bondad del ajuste GFC. Metodo ICA.');
set(gca, 'YTickLabel', colores);

clear gfc3ch2700 gfc3ch6500 gfc6ch;
clear Num Den GFCmean GFCdesv i suma;
```

Después de introducir estas instrucciones en MATLAB conviene trasladar las curvas de reflectancia espectral reconstruidas a ficheros .txt de ColorLab como se indica en la sección 2.1.11 para realizar el cálculo de los errores $\Delta E2000$. Una vez realizado esto, se calculan los errores $\Delta E2000$ como se indica dentro de 2.1.7 y después se trasladan los resultados del informe .txt generado por ColorLab a una matriz de MATLAB (deltaE) siguiendo las indicaciones de la sección 2.1.10. La matriz deltaE será

la entrada que se necesita para ejecutar las siguientes instrucciones de MATLAB orientadas a graficar los errores $\Delta E2000$ cometidos en la reconstrucción.

```
% Calculo de la media y la desviacion tipica

deltaEmean=mean(deltaE);
suma = 0;
for(i=1:5)
    suma = suma + (deltaE(i, :) - deltaEmean) .^ 2;
end
deltaEdesv = sqrt(suma / 5);
deltaE=[deltaE; deltaEmean; deltaEdesv];

% Graficar errores deltaE

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(deltaE)), 'group');
title('Errores deltaE con uno o dos iluminantes. Metodo ICA.');
set(gca, 'YTickLabel', colores);
```

5.4. Código del método CC

```
function [RMS3ch2700, RMS3ch6500, RMS6ch]=...
    metodo_CC(nParchesC)
```

```
% Los datos de entrada de esta función son SPKTtraining,
% XYZtraining2700, XYZtraining6500 y XYZtraining.

load('color_2014_05_30_cc');

% Definimos el numero de fila que ocupa cada parche dentro
% de la matriz de parches de control XYZcontrol6500 y dentro
% de la matriz elec (ver mas abajo).

A2=1;
B5=2;
D9=3;
L3=4;
M7=5;

% Tambien inicializamos la variable infinito.

infinito=99999999;

% Pasamos los parches de entrenamiento y de test al espacio Lab.

Labcontrol2700=xyz2Lab(XYZcontrol2700);
Labcontrol6500=xyz2Lab(XYZcontrol6500);
Labtraining=xyz2Lab(XYZtraining6500);

% ---------------------------------------

% Inicializamos la matriz de distancias a 2700K.
 dist2700=[];

% Calculamos la matriz dist2700. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.
```

```
for i=1:size(Labcontrol2700, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol2700(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist2700=[dist2700; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist2700 buscando las distancias mas cortas y almacenando en
% elec2700 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec2700 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec2700=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist2700, [], 2);
    for j=1:size(I, 1)
        dist2700(j, I(j))=infinito;
    end
    elec2700=[elec2700, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
```

```
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec2700(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec2700(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec2700(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec2700(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec2700(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec2700(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec2700(L3, j), :)];
```

```
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec2700(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec2700(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec2700(M7, j), :)];
end

% Reconstruccion del parche A2. Iluminante 2700K.

X3ch2700A2=pinv(XYZtrainingA2(:, 1:3))*SPKTtrainingA2;
rA23ch2700=XYZcontrol2700(A2, :)*X3ch2700A2;

% Reconstruccion del parche B5. Iluminante 2700K.

X3ch2700B5=pinv(XYZtrainingB5(:, 1:3))*SPKTtrainingB5;
rB53ch2700=XYZcontrol2700(B5, :)*X3ch2700B5;

% Reconstruccion del parche D9. Iluminante 2700K.

X3ch2700D9=pinv(XYZtrainingD9(:, 1:3))*SPKTtrainingD9;
rD93ch2700=XYZcontrol2700(D9, :)*X3ch2700D9;

% Reconstruccion del parche L3. Iluminante 2700K.

X3ch2700L3=pinv(XYZtrainingL3(:, 1:3))*SPKTtrainingL3;
rL33ch2700=XYZcontrol2700(L3, :)*X3ch2700L3;

% Reconstruccion del parche M7. Iluminante 2700K.
```

```
X3ch2700M7=pinv(XYZtrainingM7(:, 1:3))*SPKTtrainingM7;
rM73ch2700=XYZcontrol2700(M7, :)*X3ch2700M7;

% ----------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist6500=[];

% Calculamos la matriz dist6500. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol6500(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist6500=[dist6500; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist6500 buscando las distancias mas cortas y almacenando en
% elec6500 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec6500 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.
```

```
elec6500=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist6500, [], 2);
    for j=1:size(I, 1)
        dist6500(j, I(j))=infinito;
    end
    elec6500=[elec6500, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec6500(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec6500(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec6500(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec6500(B5, j), :)];
end

XYZtrainingD9=[];
```

```
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec6500(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec6500(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec6500(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec6500(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec6500(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec6500(M7, j), :)];
end


% Reconstruccion del parche A2. Iluminante 6500K.

X3ch6500A2=pinv(XYZtrainingA2(:, 4:6))*SPKTtrainingA2;
rA23ch6500=XYZcontrol6500(A2, :)*X3ch6500A2;

% Reconstruccion del parche B5. Iluminante 6500K.
```

```
X3ch6500B5=pinv(XYZtrainingB5(:, 4:6))*SPKTtrainingB5;
rB53ch6500=XYZcontrol6500(B5, :)*X3ch6500B5;

% Reconstruccion del parche D9. Iluminante 6500K.

X3ch6500D9=pinv(XYZtrainingD9(:, 4:6))*SPKTtrainingD9;
rD93ch6500=XYZcontrol6500(D9, :)*X3ch6500D9;

% Reconstruccion del parche L3. Iluminante 6500K.

X3ch6500L3=pinv(XYZtrainingL3(:, 4:6))*SPKTtrainingL3;
rL33ch6500=XYZcontrol6500(L3, :)*X3ch6500L3;

% Reconstruccion del parche M7. Iluminante 6500K.

X3ch6500M7=pinv(XYZtrainingM7(:, 4:6))*SPKTtrainingM7;
rM73ch6500=XYZcontrol6500(M7, :)*X3ch6500M7;

% ----------------------------------------

% Inicializamos la matriz de distancias con dos iluminantes.

 dist=[];

% Calculamos la matriz de distancias. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=((Labtraining(j, :)-Labcontrol2700(i, :))+...
            (Labtraining(j, :)-Labcontrol6500(i, :)));
```

```
        disth=[disth, sqrt(temp*temp')];
    end
    dist=[dist; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist buscando las distancias mas cortas y almacenando en
% elec el indice de los nParchesC parches de entrenamiento cuya
% distancia a los parches de control es menor.

% Si j es el numero de filas de elec e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist, [], 2);
    for j=1:size(I, 1)
        dist(j, I(j))=infinito;
    end
    elec=[elec, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
```

```
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec(L3, j), :)];
end

XYZtrainingM7=[];
```

```
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec(M7, j), :)];
end


% Reconstruccion del parche A2. Dos iluminantes.

X6chA2=pinv(XYZtrainingA2(:, 1:6))*SPKTtrainingA2;
rA26ch=XYZcontrol(A2, :)*X6chA2;

% Reconstruccion del parche B5. Dos iluminantes.

X6chB5=pinv(XYZtrainingB5(:, 1:6))*SPKTtrainingB5;
rB56ch=XYZcontrol(B5, :)*X6chB5;

% Reconstruccion del parche D9. Dos iluminantes.

X6chD9=pinv(XYZtrainingD9(:, 1:6))*SPKTtrainingD9;
rD96ch=XYZcontrol(D9, :)*X6chD9;

% Reconstruccion del parche L3. Dos iluminantes.

X6chL3=pinv(XYZtrainingL3(:, 1:6))*SPKTtrainingL3;
rL36ch=XYZcontrol(L3, :)*X6chL3;

% Reconstruccion del parche M7. Dos iluminantes.

X6chM7=pinv(XYZtrainingM7(:, 1:6))*SPKTtrainingM7;
rM76ch=XYZcontrol(M7, :)*X6chM7;
```

```
% ---------------------------------------

% Fundimos los resultados en tres matrices.

reconstructed3ch2700=[rA23ch2700; rB53ch2700; rD93ch2700;...
    rL33ch2700; rM73ch2700];
reconstructed3ch6500=[rA23ch6500; rB53ch6500; rD93ch6500;...
    rL33ch6500; rM73ch6500];
reconstructed6ch=[rA26ch; rB56ch; rD96ch; rL36ch; rM76ch];

% Calculo del error RMS.

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];

% Calculo de la media y la desviacion tipica.

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(suma / 5);
RMS=[RMS; RMSmean; RMSdesv];

end
```

El script que se muestra a continuación se emplea para realizar sucesivas llamadas a `metodo_CC` y sirve para dibujar una gráfica con 3 curvas, cada una de las cuales representa la suma de los errores cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, a 6500K y dos iluminantes al emplear distintas cantidades de parches de

referencia, es decir, al variar `nParchesC`. Estas curvas se muestran en la sección 3.4.

Los datos de entrada que se deben tener preparados antes de empezar a introducir estas instrucciones de MATLAB son `SPKTraining`, `XYZtraining2700`, `XYZtraining6500` y `XYZtraining`. Los datos de salida que proporciona el método son `reconstructed3ch2700`, `reconstructed3ch6500`, `reconstructed6ch`, `RMS` y `deltaE`.

```
maxN=135;


RMSan3ch2700=[];
RMSan3ch6500=[];
RMSan6ch=[];

for i=1:maxN
    [temp1, temp2, temp3]=metodo_CC(i);
    RMSan3ch2700=[RMSan3ch2700, temp1];
    RMSan3ch6500=[RMSan3ch6500, temp2];
    RMSan6ch=[RMSan6ch, temp3];
end

RMSanalisis=[sum(RMSan3ch2700); sum(RMSan3ch6500);...
    sum(RMSan6ch)];


figure;
plot([1:1:maxN], (RMSanalisis(1, :))', 'r-',...
    [1:1:maxN], (RMSanalisis(2, :))', 'g-',...
    [1:1:maxN], (RMSanalisis(3, :))', 'b-');
 xlabel('Numero de parches de referencia.');
 ylabel('Suma de errores RMS en cada reconstruccion.');
```

```
title('Analisis del numero de parches optimo de referencia.');
axis([1, maxN, 0.1, 1.1]);

clear RMSan3ch2700 RMSan3ch6500 RMSan6ch
clear temp1 temp2 temp3
clear i maxN
```

La siguiente instrucción de matlab proporcionará el valor `I` del número de parches de referencia óptimo que minimice el error cometido en la reconstrucción:

```
[C, I]=min(sum(RMSanalisis));
```

Ahora que ya se conoce el valor óptimo del número de parches de referencia se procede a realizar la reconstrucción con el siguiente conjunto de instrucciones[1]:

```
% Definimos el numero de parches cercanos con los que se va
% a realizar la reconstruccion.

nParchesC=44;

% Definimos el numero de fila que ocupa cada parche dentro
% de la matriz de parches de control XYZcontrol6500 y dentro
% de la matriz elec (ver mas abajo).
```

[1]Se inicializa la variable del número de parches cercanos a 44 porque como se ve en la sección 3.4, en las condiciones en las que se ha realizado el experimento, la suma de los todos los errores cometidos en la reconstrucción de cada uno de los parches de test con un iluminante a 2700K, otro a 6500K y dos iluminantes es mínima cuando se escogen 44 parches de referencia.

```
A2=1;
B5=2;
D9=3;
L3=4;
M7=5;

% Tambien inicializamos la variable infinito.

infinito=99999999;

% Pasamos los parches de entrenamiento y de test al espacio Lab.

Labcontrol2700=xyz2Lab(XYZcontrol2700);
Labcontrol6500=xyz2Lab(XYZcontrol6500);
Labtraining=xyz2Lab(XYZtraining6500);

% ----------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist2700=[];

% Calculamos la matriz dist2700. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol2700, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol2700(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist2700=[dist2700; disth];
```

```
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist2700 buscando las distancias mas cortas y almacenando en
% elec2700 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec2700 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec2700=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist2700, [], 2);
    for j=1:size(I, 1)
        dist2700(j, I(j))=infinito;
    end
    elec2700=[elec2700, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec2700(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
```

```
        SPKTtraining(elec2700(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec2700(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec2700(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec2700(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec2700(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec2700(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec2700(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
```

```
        XYZtraining(elec2700(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTraining(elec2700(M7, j), :)];
end

% Reconstruccion del parche A2. Iluminante 2700K.

X3ch2700A2=pinv(XYZtrainingA2(:, 1:3))*SPKTtrainingA2;
rA23ch2700=XYZcontrol2700(A2, :)*X3ch2700A2;

% Reconstruccion del parche B5. Iluminante 2700K.

X3ch2700B5=pinv(XYZtrainingB5(:, 1:3))*SPKTtrainingB5;
rB53ch2700=XYZcontrol2700(B5, :)*X3ch2700B5;

% Reconstruccion del parche D9. Iluminante 2700K.

X3ch2700D9=pinv(XYZtrainingD9(:, 1:3))*SPKTtrainingD9;
rD93ch2700=XYZcontrol2700(D9, :)*X3ch2700D9;

% Reconstruccion del parche L3. Iluminante 2700K.

X3ch2700L3=pinv(XYZtrainingL3(:, 1:3))*SPKTtrainingL3;
rL33ch2700=XYZcontrol2700(L3, :)*X3ch2700L3;

% Reconstruccion del parche M7. Iluminante 2700K.

X3ch2700M7=pinv(XYZtrainingM7(:, 1:3))*SPKTtrainingM7;
rM73ch2700=XYZcontrol2700(M7, :)*X3ch2700M7;

% ----------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist6500=[];
```

```
% Calculamos la matriz dist6500. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol6500(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist6500=[dist6500; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist6500 buscando las distancias mas cortas y almacenando en
% elec6500 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec6500 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec6500=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist6500, [], 2);
    for j=1:size(I, 1)
        dist6500(j, I(j))=infinito;
    end
```

```
    elec6500=[elec6500, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec6500(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec6500(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec6500(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec6500(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec6500(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec6500(D9, j), :)];
end
```

```
XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec6500(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec6500(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec6500(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec6500(M7, j), :)];
end


% Reconstruccion del parche A2. Iluminante 6500K.

X3ch6500A2=pinv(XYZtrainingA2(:, 4:6))*SPKTtrainingA2;
rA23ch6500=XYZcontrol6500(A2, :)*X3ch6500A2;

% Reconstruccion del parche B5. Iluminante 6500K.

X3ch6500B5=pinv(XYZtrainingB5(:, 4:6))*SPKTtrainingB5;
rB53ch6500=XYZcontrol6500(B5, :)*X3ch6500B5;

% Reconstruccion del parche D9. Iluminante 6500K.

X3ch6500D9=pinv(XYZtrainingD9(:, 4:6))*SPKTtrainingD9;
rD93ch6500=XYZcontrol6500(D9, :)*X3ch6500D9;
```

```
% Reconstruccion del parche L3. Iluminante 6500K.

X3ch6500L3=pinv(XYZtrainingL3(:, 4:6))*SPKTtrainingL3;
rL33ch6500=XYZcontrol6500(L3, :)*X3ch6500L3;

% Reconstruccion del parche M7. Iluminante 6500K.

X3ch6500M7=pinv(XYZtrainingM7(:, 4:6))*SPKTtrainingM7;
rM73ch6500=XYZcontrol6500(M7, :)*X3ch6500M7;

% ---------------------------------------

% Inicializamos la matriz de distancias con dos iluminantes.

 dist=[];

% Calculamos la matriz de distancias. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=((Labtraining(j, :)-Labcontrol2700(i, :))+...
            (Labtraining(j, :)-Labcontrol6500(i, :)));
        disth=[disth, sqrt(temp*temp')];
    end
    dist=[dist; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist buscando las distancias mas cortas y almacenando en
```

```
% elec el indice de los nParchesC parches de entrenamiento cuya
% distancia a los parches de control es menor.

% Si j es el numero de filas de elec e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist, [], 2);
    for j=1:size(I, 1)
        dist(j, I(j))=infinito;
    end
    elec=[elec, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
```

```
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec(M7, j), :)];
end
```

```
% Reconstruccion del parche A2. Dos iluminantes.

X6chA2=pinv(XYZtrainingA2(:, 1:6))*SPKTtrainingA2;
rA26ch=XYZcontrol(A2, :)*X6chA2;

% Reconstruccion del parche B5. Dos iluminantes.

X6chB5=pinv(XYZtrainingB5(:, 1:6))*SPKTtrainingB5;
rB56ch=XYZcontrol(B5, :)*X6chB5;

% Reconstruccion del parche D9. Dos iluminantes.

X6chD9=pinv(XYZtrainingD9(:, 1:6))*SPKTtrainingD9;
rD96ch=XYZcontrol(D9, :)*X6chD9;

% Reconstruccion del parche L3. Dos iluminantes.

X6chL3=pinv(XYZtrainingL3(:, 1:6))*SPKTtrainingL3;
rL36ch=XYZcontrol(L3, :)*X6chL3;

% Reconstruccion del parche M7. Dos iluminantes.

X6chM7=pinv(XYZtrainingM7(:, 1:6))*SPKTtrainingM7;
rM76ch=XYZcontrol(M7, :)*X6chM7;

% ---------------------------------------

% Fundimos los resultados en tres matrices.

reconstructed3ch2700=[rA23ch2700; rB53ch2700; rD93ch2700;...
    rL33ch2700; rM73ch2700];
reconstructed3ch6500=[rA23ch6500; rB53ch6500; rD93ch6500;...
    rL33ch6500; rM73ch6500];
```

```
reconstructed6ch=[rA26ch; rB56ch; rD96ch; rL36ch; rM76ch];

% Ploteamos los resultados

figure;
plot([380:10:730], (reconstructed3ch2700(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch2700(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch2700(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch2700(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch2700(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CC. Iluminante a 2700K.');
axis([380, 730, -0.1, 1]);

figure;
plot([380:10:730], (reconstructed3ch6500(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch6500(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch6500(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch6500(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch6500(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CC. Iluminante a 6500K.');
```

```
axis([380, 730, -0.1, 1]);

figure;
plot([380:10:730], (reconstructed6ch(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed6ch(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed6ch(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed6ch(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed6ch(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CC. Dos iluminantes.');
axis([380, 730, -0.1, 1]);


% Calculo del error RMS.

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];

% Calculo de la media y la desviacion tipica.

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(suma / 5);
```

```
RMS=[RMS; RMSmean; RMSdesv];

% Graficar errores RMS.

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(RMS)), 'group');
title('Errores RMS con uno o dos iluminantes. Metodo CC.');
set(gca, 'YTickLabel', colores);

% Limpiar el espacio de trabajo

clear A2 B5 D9 L3 M7
clear C I
clear Labcontrol2700 Labcontrol6500 Labtraining
clear RMS3ch2700 RMS3ch6500 RMS6ch RMSmean RMSdesv
clear SPKTtrainingA2 SPKTtrainingB5 SPKTtrainingD9...
    SPKTtrainingL3 SPKTtrainingM7
clear X3ch2700A2 X3ch2700B5 X3ch2700D9 X3ch2700L3...
    X3ch2700M7 X3ch6500A2 X3ch6500B5 X3ch6500D9...
    X3ch6500L3 X3ch6500M7 X6chA2 X6chB5 X6chD9 X6chL3...
    X6chM7
clear XYZtrainingA2 XYZtrainingB5 XYZtrainingD9...
    XYZtrainingL3 XYZtrainingM7
clear dist dist2700 dist6500 disth elec elec2700 elec6500
clear i j infinito
clear temp temp2
clear rA23ch2700 rB53ch2700 rD93ch2700 rL33ch2700 rM73ch2700
clear rA23ch6500 rB53ch6500 rD93ch6500 rL33ch6500 rM73ch6500
clear rA26ch rB56ch rD96ch rL36ch rM76ch

% Calcula los GFC
```

```
Num = sum((SPKTcontrol.*reconstructed3ch2700), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch2700.*reconstructed3ch2700), 2));
gfc3ch2700 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed3ch6500), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch6500.*reconstructed3ch6500), 2));
gfc3ch6500 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed6ch), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed6ch.*reconstructed6ch), 2));
gfc6ch = Num ./ Den;

gfc_full=[gfc3ch2700, gfc3ch6500, gfc6ch];

GFCmean=mean(gfc_full);
suma = 0;
for(i=1:5)
    suma = suma + (gfc_full(i, :) - GFCmean) .^ 2;
end
GFCdesv = sqrt(suma / 5);

gfc_full=[gfc_full; GFCmean; GFCdesv]

% Graficar gfc

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(gfc_full)), 'group');
title('Bondad del ajuste GFC. Metodo CC.');
set(gca, 'YTickLabel', colores);
```

```
clear gfc3ch2700 gfc3ch6500 gfc6ch;
clear Num Den GFCmean GFCdesv i suma;
```

Después de introducir estas instrucciones en MATLAB conviene trasladar las curvas de reflectancia espectral reconstruidas a ficheros .txt de ColorLab como se indica en la sección 2.1.11 para realizar el cálculo de los errores $\Delta E2000$. Una vez realizado esto, se calculan los errores $\Delta E2000$ como se indica dentro de 2.1.7 y después se trasladan los resultados del informe .txt generado por ColorLab a una matriz de MATLAB (deltaE) siguiendo las indicaciones de la sección 2.1.10. La matriz deltaE será la entrada que se necesita para ejecutar las siguientes instrucciones de MATLAB orientadas a graficar los errores $\Delta E2000$ cometidos en la reconstrucción.

```
% Calculo de la media y la desviacion tipica

deltaEmean=mean(deltaE);
suma = 0;
for(i=1:5)
    suma = suma + (deltaE(i, :) - deltaEmean) .^ 2;
end
deltaEdesv = sqrt(suma / 5);
deltaE=[deltaE; deltaEmean; deltaEdesv];

% Graficar errores deltaE

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(deltaE)), 'group');
title('Errores deltaE con uno o dos iluminantes. Metodo CC.');
```

```
set(gca, 'YTickLabel', colores);
```

5.5. Código del método CCC

La función cuadratizar transforma una matriz de valores XYZ ordenados en filas en otra matriz que contiene 3 columnas adicionales con los productos cruzados y 3 columnas más con los cuadrados de los valores XYZ.

```
function [X]=cuadratizar(Y);

X=[Y, (Y(:, 1).*Y(:, 2)), (Y(:, 1).*Y(:, 3)),...
    (Y(:, 2).*Y(:, 3)), (Y(:, 1).*Y(:, 1)),...
    (Y(:, 2).*Y(:, 2)), (Y(:, 3).*Y(:, 3))];

end
```

La función `metodo_CCC` se parece mucho a la función mencionada en 2.5.

```
function [RMS3ch2700, RMS3ch6500, RMS6ch]=...
    metodo_CCC(nParchesC);

% Los datos de entrada de esta funci?n son SPKTtraining,
% XYZtraining2700, XYZtraining6500 y XYZtraining.

load('color_2014_06_01_ccc');
```

```
% Definimos el numero de fila que ocupa cada parche dentro
% de la matriz de parches de control XYZcontrol6500 y dentro
% de la matriz elec (ver mas abajo).

A2=1;
B5=2;
D9=3;
L3=4;
M7=5;

% Tambien inicializamos la variable infinito.

infinito=99999999;

% Pasamos los parches de entrenamiento y de test al espacio Lab.

Labcontrol2700=xyz2Lab(XYZcontrol2700);
Labcontrol6500=xyz2Lab(XYZcontrol6500);
Labtraining=xyz2Lab(XYZtraining6500);

% ----------------------------------------

% Inicializamos la matriz de distancias a 2700K.
 dist2700=[];

% Calculamos la matriz dist2700. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol2700, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol2700(i, :));
```

```
        disth=[disth, sqrt(temp*temp')];
    end
    dist2700=[dist2700; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist2700 buscando las distancias mas cortas y almacenando en
% elec2700 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec2700 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec2700=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist2700, [], 2);
    for j=1:size(I, 1)
        dist2700(j, I(j))=infinito;
    end
    elec2700=[elec2700, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
```

```
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec2700(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec2700(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec2700(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec2700(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec2700(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec2700(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec2700(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec2700(L3, j), :)];
end

XYZtrainingM7=[];
```

```
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec2700(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec2700(M7, j), :)];
end

% Reconstruccion del parche A2. Iluminante 2700K.

X3ch2700A2=pinv(cuadratizar(XYZtrainingA2(:, 1:3)))*...
    SPKTtrainingA2;
rA23ch2700=cuadratizar(XYZcontrol2700(A2, :))*X3ch2700A2;

% Reconstruccion del parche B5. Iluminante 2700K.

X3ch2700B5=pinv(cuadratizar(XYZtrainingB5(:, 1:3)))*...
    SPKTtrainingB5;
rB53ch2700=cuadratizar(XYZcontrol2700(B5, :))*X3ch2700B5;

% Reconstruccion del parche D9. Iluminante 2700K.

X3ch2700D9=pinv(cuadratizar(XYZtrainingD9(:, 1:3)))*...
    SPKTtrainingD9;
rD93ch2700=cuadratizar(XYZcontrol2700(D9, :))*X3ch2700D9;

% Reconstruccion del parche L3. Iluminante 2700K.

X3ch2700L3=pinv(cuadratizar(XYZtrainingL3(:, 1:3)))*...
    SPKTtrainingL3;
rL33ch2700=cuadratizar(XYZcontrol2700(L3, :))*X3ch2700L3;

% Reconstruccion del parche M7. Iluminante 2700K.
```

```
X3ch2700M7=pinv(cuadratizar(XYZtrainingM7(:, 1:3)))*...
    SPKTtrainingM7;
rM73ch2700=cuadratizar(XYZcontrol2700(M7, :))*X3ch2700M7;

% ----------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist6500=[];

% Calculamos la matriz dist6500. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol6500(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist6500=[dist6500; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist6500 buscando las distancias mas cortas y almacenando en
% elec6500 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec6500 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.
```

```
elec6500=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist6500, [], 2);
    for j=1:size(I, 1)
        dist6500(j, I(j))=infinito;
    end
    elec6500=[elec6500, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec6500(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec6500(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec6500(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec6500(B5, j), :)];
end

XYZtrainingD9=[];
```

```
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec6500(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec6500(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec6500(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec6500(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec6500(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec6500(M7, j), :)];
end


% Reconstruccion del parche A2. Iluminante 6500K.

X3ch6500A2=pinv(cuadratizar(XYZtrainingA2(:, 4:6)))*...
    SPKTtrainingA2;
rA23ch6500=cuadratizar(XYZcontrol6500(A2, :))*X3ch6500A2;

% Reconstruccion del parche B5. Iluminante 6500K.
```

```
X3ch6500B5=pinv(cuadratizar(XYZtrainingB5(:, 4:6)))*...
    SPKTtrainingB5;
rB53ch6500=cuadratizar(XYZcontrol6500(B5, :))*X3ch6500B5;

% Reconstruccion del parche D9. Iluminante 6500K.

X3ch6500D9=pinv(cuadratizar(XYZtrainingD9(:, 4:6)))*...
    SPKTtrainingD9;
rD93ch6500=cuadratizar(XYZcontrol6500(D9, :))*X3ch6500D9;

% Reconstruccion del parche L3. Iluminante 6500K.

X3ch6500L3=pinv(cuadratizar(XYZtrainingL3(:, 4:6)))*...
    SPKTtrainingL3;
rL33ch6500=cuadratizar(XYZcontrol6500(L3, :))*X3ch6500L3;

% Reconstruccion del parche M7. Iluminante 6500K.

X3ch6500M7=pinv(cuadratizar(XYZtrainingM7(:, 4:6)))*...
    SPKTtrainingM7;
rM73ch6500=cuadratizar(XYZcontrol6500(M7, :))*X3ch6500M7;

% ----------------------------------------

% Inicializamos la matriz de distancias con dos iluminantes.

 dist=[];

% Calculamos la matriz de distancias. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.
```

```
for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=((Labtraining(j, :)-Labcontrol2700(i, :))+...
            (Labtraining(j, :)-Labcontrol6500(i, :)));
        disth=[disth, sqrt(temp*temp')];
    end
    dist=[dist; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist buscando las distancias mas cortas y almacenando en
% elec el indice de los nParchesC parches de entrenamiento cuya
% distancia a los parches de control es menor.

% Si j es el numero de filas de elec e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist, [], 2);
    for j=1:size(I, 1)
        dist(j, I(j))=infinito;
    end
    elec=[elec, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
```

```
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec(L3, j), :)];
```

```
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec(M7, j), :)];
end


% Reconstruccion del parche A2. Dos iluminantes.

X6chA2=pinv(cuadratizar(XYZtrainingA2(:, 1:6)))*...
    SPKTtrainingA2;
rA26ch=cuadratizar(XYZcontrol(A2, :))*X6chA2;

% Reconstruccion del parche B5. Dos iluminantes.

X6chB5=pinv(cuadratizar(XYZtrainingB5(:, 1:6)))*...
    SPKTtrainingB5;
rB56ch=cuadratizar(XYZcontrol(B5, :))*X6chB5;

% Reconstruccion del parche D9. Dos iluminantes.

X6chD9=pinv(cuadratizar(XYZtrainingD9(:, 1:6)))*...
    SPKTtrainingD9;
rD96ch=cuadratizar(XYZcontrol(D9, :))*X6chD9;

% Reconstruccion del parche L3. Dos iluminantes.
```

```
X6chL3=pinv(cuadratizar(XYZtrainingL3(:, 1:6)))*...
    SPKTtrainingL3;
rL36ch=cuadratizar(XYZcontrol(L3, :))*X6chL3;

% Reconstruccion del parche M7. Dos iluminantes.

X6chM7=pinv(cuadratizar(XYZtrainingM7(:, 1:6)))*...
    SPKTtrainingM7;
rM76ch=cuadratizar(XYZcontrol(M7, :))*X6chM7;

% ---------------------------------------

% Fundimos los resultados en tres matrices.

reconstructed3ch2700=[rA23ch2700; rB53ch2700; rD93ch2700;...
    rL33ch2700; rM73ch2700];
reconstructed3ch6500=[rA23ch6500; rB53ch6500; rD93ch6500;...
    rL33ch6500; rM73ch6500];
reconstructed6ch=[rA26ch; rB56ch; rD96ch; rL36ch; rM76ch];

% Calculo del error RMS.

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];

% Calculo de la media y la desviacion tipica.

RMSmean=mean(RMS);
suma = 0;
for(i=1:5)
    suma = suma + (RMS(i, :) - RMSmean) .^ 2;
end
```

```
RMSdesv = sqrt(suma / 5);
RMS=[RMS; RMSmean; RMSdesv];

end
```

El script que se muestra a continuación se emplea para realizar sucesivas llamadas a `metodo_CCC` y sirve para dibujar una gráfica con 3 curvas, cada una de las cuales representa la suma de los errores cometidos en la reconstrucción de los parches de test con un iluminante a 2700K, a 6500K y dos iluminantes al emplear distintas cantidades de parches de referencia, es decir, al variar `nParchesC`. Estas curvas se muestran en la sección 3.5.

Los datos de entrada que se deben tener preparados antes de empezar a introducir estas instrucciones de MATLAB son `SPKTraining`, `XYZtraining2700`, `XYZtraining6500` y `XYZtraining`. Los datos de salida que proporciona el método son `reconstructed3ch2700`, `reconstructed3ch6500`, `reconstructed6ch`, `RMS` y `deltaE`.

```
maxN=135;


RMSan3ch2700=[];
RMSan3ch6500=[];
RMSan6ch=[];

for i=1:maxN
    [temp1, temp2, temp3]=metodo_CCC(i);
    RMSan3ch2700=[RMSan3ch2700, temp1];
    RMSan3ch6500=[RMSan3ch6500, temp2];
    RMSan6ch=[RMSan6ch, temp3];
end
```

```
RMSanalisis=[sum(RMSan3ch2700); sum(RMSan3ch6500); sum(RMSan6ch)]
    ;


figure;
plot([1:1:maxN], (RMSanalisis(1, :))', 'r-',...
    [1:1:maxN], (RMSanalisis(2, :))', 'g-',...
    [1:1:maxN], (RMSanalisis(3, :))', 'b-');
 xlabel('Numero de parches de referencia.');
 ylabel('Suma de errores RMS en cada reconstruccion.');
 title('Numero de parches optimo de referencia. Metodo CCC.');
 axis([1, maxN, 0, 7]);

 clear RMSan3ch2700 RMSan3ch6500 RMSan6ch
 clear temp1 temp2 temp3
 clear i maxN
```

La siguiente instrucción de matlab proporcionará el valor `I` del número de parches de referencia óptimo que minimice el error cometido en la reconstrucción:

```
[C, I]=min(sum(RMSanalisis));
```

Ahora que ya se conoce el valor óptimo del número de parches de referencia se procede a realizar la reconstrucción con el siguiente conjunto de instrucciones[2]:

[2]Se inicializa la variable del número de parches cercanos a 82 porque como se ve en la sección 3.5, en las condiciones en las que se ha realizado el experimento, la suma de los todos los errores cometidos en la reconstrucción de cada uno de los parches de test con un iluminante a 2700K, otro a 6500K y dos iluminantes es mínima cuando se escogen 82 parches de referencia.

```
% Definimos el numero de parches cercanos con los que se va
% a realizar la reconstruccion.

nParchesC=82;

% Definimos el numero de fila que ocupa cada parche dentro
% de la matriz de parches de control XYZcontrol6500 y dentro
% de la matriz elec (ver mas abajo).

A2=1;
B5=2;
D9=3;
L3=4;
M7=5;

% Tambien inicializamos la variable infinito.

infinito=99999999;

% Pasamos los parches de entrenamiento y de test al espacio Lab.

Labcontrol2700=xyz2Lab(XYZcontrol2700);
Labcontrol6500=xyz2Lab(XYZcontrol6500);
Labtraining=xyz2Lab(XYZtraining6500);

% ---------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist2700=[];

% Calculamos la matriz dist2700. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
```

```
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol2700, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol2700(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist2700=[dist2700; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist2700 buscando las distancias mas cortas y almacenando en
% elec2700 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec2700 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec2700=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist2700, [], 2);
    for j=1:size(I, 1)
        dist2700(j, I(j))=infinito;
    end
    elec2700=[elec2700, I];
end
```

```
% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec2700(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec2700(A2, j), :)];
end

XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec2700(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec2700(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec2700(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec2700(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
```

```
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec2700(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec2700(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec2700(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec2700(M7, j), :)];
end

% Reconstruccion del parche A2. Iluminante 2700K.

X3ch2700A2=pinv(cuadratizar(XYZtrainingA2(:, 1:3)))*...
    SPKTtrainingA2;
rA23ch2700=cuadratizar(XYZcontrol2700(A2, :))*...
    X3ch2700A2;

% Reconstruccion del parche B5. Iluminante 2700K.

X3ch2700B5=pinv(cuadratizar(XYZtrainingB5(:, 1:3)))*...
    SPKTtrainingB5;
rB53ch2700=cuadratizar(XYZcontrol2700(B5, :))*X3ch2700B5;

% Reconstruccion del parche D9. Iluminante 2700K.

X3ch2700D9=pinv(cuadratizar(XYZtrainingD9(:, 1:3)))*...
    SPKTtrainingD9;
rD93ch2700=cuadratizar(XYZcontrol2700(D9, :))*...
    X3ch2700D9;
```

```
% Reconstruccion del parche L3. Iluminante 2700K.

X3ch2700L3=pinv(cuadratizar(XYZtrainingL3(:, 1:3)))*...
    SPKTtrainingL3;
rL33ch2700=cuadratizar(XYZcontrol2700(L3, :))*...
    X3ch2700L3;

% Reconstruccion del parche M7. Iluminante 2700K.

X3ch2700M7=pinv(cuadratizar(XYZtrainingM7(:, 1:3)))*...
    SPKTtrainingM7;
rM73ch2700=cuadratizar(XYZcontrol2700(M7, :))*...
    X3ch2700M7;

% ----------------------------------------

% Inicializamos la matriz de distancias a 6500K.
 dist6500=[];

% Calculamos la matriz dist6500. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=(Labtraining(j, :)-Labcontrol6500(i, :));
        disth=[disth, sqrt(temp*temp')];
    end
    dist6500=[dist6500; disth];
end
```

```
% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist6500 buscando las distancias mas cortas y almacenando en
% elec6500 el indice de los nParchesC parches de entrenamiento
% cuya distancia a los parches de control es menor.

% Si j es el numero de filas de elec6500 e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec6500=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist6500, [], 2);
    for j=1:size(I, 1)
        dist6500(j, I(j))=infinito;
    end
    elec6500=[elec6500, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTtrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec6500(A2, j), :)];
    SPKTtrainingA2=[SPKTtrainingA2;...
        SPKTtraining(elec6500(A2, j), :)];
end
```

```
XYZtrainingB5=[];
SPKTtrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec6500(B5, j), :)];
    SPKTtrainingB5=[SPKTtrainingB5;...
        SPKTtraining(elec6500(B5, j), :)];
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec6500(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec6500(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec6500(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec6500(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec6500(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
```

```
        SPKTtraining(elec6500(M7, j), :)];
end


% Reconstruccion del parche A2. Iluminante 6500K.

X3ch6500A2=pinv(cuadratizar(XYZtrainingA2(:, 4:6)))*...
    SPKTtrainingA2;
rA23ch6500=cuadratizar(XYZcontrol6500(A2, :))*...
    X3ch6500A2;

% Reconstruccion del parche B5. Iluminante 6500K.

X3ch6500B5=pinv(cuadratizar(XYZtrainingB5(:, 4:6)))*...
    SPKTtrainingB5;
rB53ch6500=cuadratizar(XYZcontrol6500(B5, :))*...
    X3ch6500B5;

% Reconstruccion del parche D9. Iluminante 6500K.

X3ch6500D9=pinv(cuadratizar(XYZtrainingD9(:, 4:6)))*...
    SPKTtrainingD9;
rD93ch6500=cuadratizar(XYZcontrol6500(D9, :))*...
    X3ch6500D9;

% Reconstruccion del parche L3. Iluminante 6500K.

X3ch6500L3=pinv(cuadratizar(XYZtrainingL3(:, 4:6)))*...
    SPKTtrainingL3;
rL33ch6500=cuadratizar(XYZcontrol6500(L3, :))*...
    X3ch6500L3;

% Reconstruccion del parche M7. Iluminante 6500K.
```

```
X3ch6500M7=pinv(cuadratizar(XYZtrainingM7(:, 4:6)))*...
    SPKTtrainingM7;
rM73ch6500=cuadratizar(XYZcontrol6500(M7, :))*...
    X3ch6500M7;

% ----------------------------------------

% Inicializamos la matriz de distancias con dos iluminantes.

 dist=[];

% Calculamos la matriz de distancias. Si i son las filas de la
% matriz de distancias y j son las columnas de dicha matriz, el
% elemento dist(i, j) contiene la distancia medida en deltaE76
% entre el parche de control i y el parche de entrenamiento j.

for i=1:size(Labcontrol6500, 1)
    disth=[];
    for j=1:size(Labtraining, 1)
        temp=((Labtraining(j, :)-Labcontrol2700(i, :))+...
            (Labtraining(j, :)-Labcontrol6500(i, :)));
        disth=[disth, sqrt(temp*temp')];
    end
    dist=[dist; disth];
end

% Ahora vamos a recorrer tantas veces como canales se usan
% para la reconstruccion cada una de las filas de la matriz
% dist buscando las distancias mas cortas y almacenando en
% elec el indice de los nParchesC parches de entrenamiento cuya
% distancia a los parches de control es menor.

% Si j es el numero de filas de elec e i es el numero de
% columnas, en cada fila j se almacena en orden creciente de
```

```
% columna a columna el indice de los parches de entrenamiento
% cuya distancia a los parches de control es menor.

elec=[];
temp2=[];

for i=1:nParchesC
    [C, I]=min(dist, [], 2);
    for j=1:size(I, 1)
        dist(j, I(j))=infinito;
    end
    elec=[elec, I];
end

% Ahora vamos a almacenar dentro de varias matrices, los
% valores parches que se van a usar para realizar para el
% entrenamiento.

XYZtrainingA2=[];
SPKTrainingA2=[];
for j=1:nParchesC
    XYZtrainingA2=[XYZtrainingA2;...
        XYZtraining(elec(A2, j), :)];
    SPKTrainingA2=[SPKTrainingA2;...
        SPKTraining(elec(A2, j), :)];
end

XYZtrainingB5=[];
SPKTrainingB5=[];
for j=1:nParchesC
    XYZtrainingB5=[XYZtrainingB5;...
        XYZtraining(elec(B5, j), :)];
    SPKTrainingB5=[SPKTrainingB5;...
        SPKTraining(elec(B5, j), :)];
```

```
end

XYZtrainingD9=[];
SPKTtrainingD9=[];
for j=1:nParchesC
    XYZtrainingD9=[XYZtrainingD9;...
        XYZtraining(elec(D9, j), :)];
    SPKTtrainingD9=[SPKTtrainingD9;...
        SPKTtraining(elec(D9, j), :)];
end

XYZtrainingL3=[];
SPKTtrainingL3=[];
for j=1:nParchesC
    XYZtrainingL3=[XYZtrainingL3;...
        XYZtraining(elec(L3, j), :)];
    SPKTtrainingL3=[SPKTtrainingL3;...
        SPKTtraining(elec(L3, j), :)];
end

XYZtrainingM7=[];
SPKTtrainingM7=[];
for j=1:nParchesC
    XYZtrainingM7=[XYZtrainingM7;...
        XYZtraining(elec(M7, j), :)];
    SPKTtrainingM7=[SPKTtrainingM7;...
        SPKTtraining(elec(M7, j), :)];
end


% Reconstruccion del parche A2. Dos iluminantes.

X6chA2=pinv(cuadratizar(XYZtrainingA2(:, 1:6)))*...
    SPKTtrainingA2;
```

```
rA26ch=cuadratizar(XYZcontrol(A2, :))*X6chA2;

% Reconstruccion del parche B5. Dos iluminantes.

X6chB5=pinv(cuadratizar(XYZtrainingB5(:, 1:6)))*...
    SPKTtrainingB5;
rB56ch=cuadratizar(XYZcontrol(B5, :))*X6chB5;

% Reconstruccion del parche D9. Dos iluminantes.

X6chD9=pinv(cuadratizar(XYZtrainingD9(:, 1:6)))*...
    SPKTtrainingD9;
rD96ch=cuadratizar(XYZcontrol(D9, :))*X6chD9;

% Reconstruccion del parche L3. Dos iluminantes.

X6chL3=pinv(cuadratizar(XYZtrainingL3(:, 1:6)))*...
    SPKTtrainingL3;
rL36ch=cuadratizar(XYZcontrol(L3, :))*X6chL3;

% Reconstruccion del parche M7. Dos iluminantes.

X6chM7=pinv(cuadratizar(XYZtrainingM7(:, 1:6)))*...
    SPKTtrainingM7;
rM76ch=cuadratizar(XYZcontrol(M7, :))*X6chM7;

% ---------------------------------------

% Fundimos los resultados en tres matrices.

reconstructed3ch2700=[rA23ch2700; rB53ch2700; rD93ch2700;...
    rL33ch2700; rM73ch2700];
reconstructed3ch6500=[rA23ch6500; rB53ch6500; rD93ch6500;...
    rL33ch6500; rM73ch6500];
```

```
reconstructed6ch=[rA26ch; rB56ch; rD96ch; rL36ch; rM76ch];

% Ploteamos los resultados

figure;
plot([380:10:730], (reconstructed3ch2700(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch2700(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch2700(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch2700(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch2700(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CCC. Iluminante a 2700K.');
axis([380, 730, -0.1, 1]);

figure;
plot([380:10:730], (reconstructed3ch6500(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed3ch6500(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed3ch6500(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed3ch6500(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed3ch6500(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CCC. Iluminante a 6500K.');
```

```
axis([380, 730, -0.1, 1]);

figure;
plot([380:10:730], (reconstructed6ch(1, :))', 'k-',...
    [380:10:730], (SPKTcontrol(1, :))', 'k--',...
    [380:10:730], (reconstructed6ch(2, :))', 'b-',...
    [380:10:730], (SPKTcontrol(2, :))', 'b--',...
    [380:10:730], (reconstructed6ch(3, :))', 'g-',...
    [380:10:730], (SPKTcontrol(3, :))', 'g--',...
    [380:10:730], (reconstructed6ch(4, :))', 'r-',...
    [380:10:730], (SPKTcontrol(4, :))', 'r--',...
    [380:10:730], (reconstructed6ch(5, :))', 'y-',...
    [380:10:730], (SPKTcontrol(5, :))', 'y--');
xlabel('Longitud de onda (nm)');
ylabel('Reflectancia espectral');
title('Reconstruccion metodo CCC. Dos iluminantes.');
axis([380, 730, -0.1, 1]);


% Calculo del error RMS.

RMS3ch2700=rms(reconstructed3ch2700, SPKTcontrol);
RMS3ch6500=rms(reconstructed3ch6500, SPKTcontrol);
RMS6ch=rms(reconstructed6ch, SPKTcontrol);
RMS=[RMS3ch2700 RMS3ch6500 RMS6ch];

% Calculo de la media y la desviacion tipica.

RMSmean=mean(RMS);
sum = 0;
for(i=1:5)
    sum = sum + (RMS(i, :) - RMSmean) .^ 2;
end
RMSdesv = sqrt(sum / 5);
```

```
RMS=[RMS; RMSmean; RMSdesv];

% Graficar errores RMS.

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(RMS)), 'group');
title('Errores RMS con uno o dos iluminantes. Metodo CCC.');
set(gca, 'YTickLabel', colores);

% Limpiar el espacio de trabajo

clear A2 B5 D9 L3 M7
clear C I
clear Labcontrol2700 Labcontrol6500 Labtraining
clear RMS3ch2700 RMS3ch6500 RMS6ch RMSmean RMSdesv
clear SPKTtrainingA2 SPKTtrainingB5 SPKTtrainingD9...
    SPKTtrainingL3 SPKTtrainingM7
clear X3ch2700A2 X3ch2700B5 X3ch2700D9 X3ch2700L3...
    X3ch2700M7 X3ch6500A2 X3ch6500B5 X3ch6500D9...
    X3ch6500L3 X3ch6500M7 X6chA2 X6chB5 X6chD9 X6chL3...
    X6chM7
clear XYZtrainingA2 XYZtrainingB5 XYZtrainingD9...
    XYZtrainingL3 XYZtrainingM7
clear dist dist2700 dist6500 disth elec elec2700 elec6500
clear i j infinito
clear temp temp2
clear rA23ch2700 rB53ch2700 rD93ch2700 rL33ch2700 rM73ch2700
clear rA23ch6500 rB53ch6500 rD93ch6500 rL33ch6500 rM73ch6500
clear rA26ch rB56ch rD96ch rL36ch rM76ch

% Calcula los GFC
```

```
Num = sum((SPKTcontrol.*reconstructed3ch2700), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch2700.*reconstructed3ch2700), 2));
gfc3ch2700 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed3ch6500), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed3ch6500.*reconstructed3ch6500), 2));
gfc3ch6500 = Num ./ Den;

Num = sum((SPKTcontrol.*reconstructed6ch), 2);
Den = sqrt(sum((SPKTcontrol.*SPKTcontrol), 2)) .*...
 sqrt(sum((reconstructed6ch.*reconstructed6ch), 2));
gfc6ch = Num ./ Den;

gfc_full=[gfc3ch2700, gfc3ch6500, gfc6ch];

GFCmean=mean(gfc_full);
suma = 0;
for(i=1:5)
    suma = suma + (gfc_full(i, :) - GFCmean) .^ 2;
end
GFCdesv = sqrt(suma / 5);

gfc_full=[gfc_full; GFCmean; GFCdesv]

% Graficar gfc

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(gfc_full)), 'group');
title('Bondad del ajuste GFC. Metodo CC.');
set(gca, 'YTickLabel', colores);
```

```
clear gfc3ch2700 gfc3ch6500 gfc6ch;
clear Num Den GFCmean GFCdesv i suma;
```

Después de introducir estas instrucciones en MATLAB conviene trasladar las curvas de reflectancia espectral reconstruidas a ficheros .txt de ColorLab como se indica en la sección 2.1.11 para realizar el cálculo de los errores $\Delta E2000$. Una vez realizado esto, se calculan los errores $\Delta E2000$ como se indica dentro de 2.1.7 y después se trasladan los resultados del informe .txt generado por ColorLab a una matriz de MATLAB (deltaE) siguiendo las indicaciones de la sección 2.1.10. La matriz deltaE será la entrada que se necesita para ejecutar las siguientes instrucciones de MATLAB orientadas a graficar los errores $\Delta E2000$ cometidos en la reconstrucción.

```
% Calculo de la media y la desviacion tipica

deltaEmean=mean(deltaE);
sum = 0;
for(i=1:5)
    sum = sum + (deltaE(i, :) - deltaEmean) .^ 2;
end
deltaEdesv = sqrt(sum / 5);
deltaE=[deltaE; deltaEmean; deltaEdesv];

% Graficar errores deltaE

colores={'DESV. TIP.', 'MEDIA', 'Amarillo M7',...
    'Rojo L3', 'Verde D9', 'Azul B5', 'Neutro A2'};
figure;
barh(fliplr(flipud(deltaE)), 'group');
title('Errores deltaE con uno o dos iluminantes. Metodo CCC.');
```

```
set(gca, 'YTickLabel', colores);
```

5.6. Especificaciones de la cámara empleada

Las especificaciones completas de la cámara empleadas se encuentran disponibles en la página web https://www.canon.es/. A continuación se muestra un pequeño resumen de las mismas:

GENERAL
Tipo de cámara: réflex, objetivos intercambiables
Fecha de presentación: febrero, 2010

CAPTACIÓN DE LA IMAGEN
Tamaño del sensor: 22,3 x 14,9 mm
Resolución total: 18,7 MP
Resolución efectiva: 18 MP
Espacio de color: sRGB, Adobe RGB
Filtro de color: RGB

ÓPTICA
Montura Canon EF-S
Factor de recorte: 1.6x

CONTROL DE LA EXPOSICIÓN
Modo manual y automático
Prioridad a la obturación
Prioridad a la apertura
Programa
Escenas
Compensación de exposición de -5 a +5 EV
Pasos 1/2, 1/3
Velocidad de obturación máxima: 1/4.000s
Velocidad de obturación mínima: 30 s

Modo B
Sensibilidad 100-12.800 ISO
Medición ponderada, puntual, matricial y parcial

CONTROL DEL DISPARO
Disparo simple
Archivos JPEG, RAW
Resolucion máxima (en píxeles) 2.592 x 1.728
Disparo en ráfaga 3,7 fps
Vídeo .MOV

VISUALIZACIÓN DE LA IMAGEN
Visor réflex
Cobertura 95 %
Monitor TFT 3 pulgadas
Resolución del monitor de 1.040.000 píxeles

CUERPO
Sumergible
Peso 530 g
Dimensiones 128,8 mm x 97,3 mm x 62 mm

5.7. Espeficicaciones del espectrofotómetro empleado

El espectrofotómetro empleado es el XRite Eye One Pro y sus especificaciones técnicas se encuentran publicadas en http://www.xrite.com/. El dispositivo es un aparato de reducidas dimensiones que realiza mediciones de energía emisivas o reflexivas en el rango de 380 a 730 nm a intervalos de 10 nm. Se puede emplear también para calibrar monitores.

Bibliografía

[1] Hunter, DiCarlo, and Pollard. Six color scanning. *Technical Papers of 4th European Conference on Colour in Graphics, Imaging, and Vision / 10th International Symposium on Multispectral Colour Science*, (June 2008):570–574, 2008. URL http://www.proceedings.com/04561.html.

[2] PD Burns DY Tzeng RS Berns, FH Imai. Electronic imaging: Processing, printing, and publishing in color. *SPIE Proceedings. Multispectral-Based Color Reproduction Research at the Munsell Color Science Laboratory*, (3409):14–25, 1998.

[3] RS Berns. Challenges for color science in multimedia imaging. derby, uk. *CIM'98: Colour Imaging in Multimedia*, pages 123–133, 1998.

[4] Y Komiya H Fukuda H Hanelshi M Yamaguchi N Ohyama K Ohsawa, T Ajito. Six band hdtv camera system for spectrum-based color reproduction. *Journal of the Imaging Science and Technology*, (48):85–92, 2004.

[5] F Zaraga G Langfelder, AF Longoni. Implementation of a multispectral color imaging device without color filter array. *Digital Photography VII*, (7876-7877 of Proceedings of SPIE / International Science and Technology Electronic Imaging), 2011.

[6] MD Grossberg SK Nayar J Park, M Lee. Multispectral imaging using multiplexed illumination. *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[7] Pujol J. Herrera-Ramirez J, Vilaseca M. Portable multispectral imaging systen based on light emitting diodes for spectral recovery from 370 to 1630 nm. *Applied Optics*, (53), 2014.

[8] Jaume Pujol Jorge Herrera-Ramirez, Meritxell Vilaseca. Diseño e implementacion de un sistema multiespectral en el rango ultravioleta, visible e infrarrojo. aplicacion al estudio y conservacion de obras de arte. thesis, Universidad Politecnica de Catalunya, 2014. URL http://www.tdx.cat/handle/10803/277543.

[9] Francisco J. Burgos Lidia Font Rosa Senserrich Jaume Pujol Jorge Herrera-Ramirez, Meritxell Vilaseca. Artwork imaging from 370 to 1630 nm using a novel multispectral system based on light emitting diodes. *Color Research and Application*, (53), 2014.

[10] A. Hunter and S. Pollard. Generating a scanned image using two light sources having different frequency spectra, published as patent application. (GB2437577A), 2007.

[11] E. Perales M. Vilaseca J. Pujol F. Martinez-Verdu, E. Chorro. *Colour Measurement, Chapter 6, Principles, Advances and Industrial Applications. A volume in Woodhead Publishing Series in Textiles.* M.L. Gulrajani, 2010.

[12] Test report color reproduction of consumer flatbed scanners. *Image Engineering Dietmar Wueller*, 2007. URL http://digitalkamera.image-engineering.de.

[13] J.C. Russ. *The image processing handbook*. CRC Press, Boca Raton, USA, 1995.

[14] B. Olding U. Barnhofer, J. DiCarlo and B. A. Wandell. Color estimation error trade-offs. *Proc. SPIE*, (5017), 2003.

[15] Clara Plata. Caracterizacion multiespectral de objetos con textura empleando propiedades invariantes frente a los cambios de iluminacion. thesis, Universidad de Granada, Facultad de Ciencias, Departamento de Optica, 2009. URL http://www.tdx.cat/handle/10803/17003.

[16] JY Hardeberg. Filter selection for multispectral color image acquisition. *Journal of Imaging Science and Technology*, (48):105–110, 2004.

[17] JY Hardeberg D Connah, A Alsam. Multispectral imaging: how many sensors do we need? *Journal of Imaging Science and Technology*, (50):45–52, 2006.

[18] H Brettel JP Crettez H Matre JY Hardeberg, F Schmitt. Colour imaging in multimedia, (multispectral imaging in multimedia, 1998),. *Proceedings, CIM'98*, pages 75–86, 1998.

[19] M Dupouy P Cotte. *PICS, (CRISATEL High Resolution Multispectral System)*. International Science and Technology, 2003.

[20] Day EA Imai FH, Taplin LA. Comparative study of spectral reflectance estimation based on broad-band imaging systems. as part of end-to-end color reproduction from scene to reproduction using mvsi. 2003.

[21] J. Y. Hardeberg R. Shrestha, A. Mansouri. *Journal on Advances in Signal Processing*, (57):1–15, 2011.

[22] Valero EM Romero J Hernandez-Andres J, Nieves JL. Spectral daylight recovery using only a few sensors. *Journal of the Optical Society of America*, (21):13–23, 2004.

[23] Li C Hardeberg JY Connah D. Cheung V, Westland S. Characterization of trichromatic color cameras by using a new multispectral imaging technique. *Journal of the Optical Society of America*, (22):1231–1240, 2005.

[24] Hardeberg JY. Acquisition and reproduction of colored images: Colorimetric and multispectral approaches. thesis, Ecole Superieur Nationale des Telecommunications, 1999.

[25] FH Imai and RS Berns. Spectral estimation using trichromatic digital cameras. *Multispectral Imaging and Color Reproduction for Digital Archives*, 1999. URL http://www.cis.rit.edu/research/mcsl2/online/Spectral/TechnicalPapers/SpecEstimRGBCamera_Chiba99.pdf.

[26] Clara Plata and EM Valero. Supervised training sample selection for the estimation of spectral reflectance using a rgb camera. *Conference on Color in Graphics, Imaging and Vision, CGIV 2008*, pages 519–522, 2008. URL http://www.ingentaconnect.com/content/ist/cgiv/2008/00002008/00000001/art00112.

[27] R Schettini G Novati, P Pellegri. Selection of filters for multispectral acquisition using the filter vectors analysis method. *Color Imaging IX: Processing, Hardcopy, and Applications*, (5293 of SPIE Proceedings):20–26, 2004.

[28] DH Foster SMC Nascimento, FP Ferreira. Statistics of spatial cone-excitation ratios in natural scenes. *Journal of the Optical Society of America*, (19):1484–1490, 2002.

[29] Hans Brettel. Multispectral image capture across the web. *7th Annual Color and Imaging Conference*, (4):8–10, 1999. URL http://www.ingentaconnect.com/content/ist/cic/1999/00001999/00000001/art00060.

[30] Brettel H. Hardeberg JY, Schmitt F. Multispectral color image capture using a liquid crystal tunable filter. *Optical Engineering*, (41):2532–2548, 2002.

[31] CC Hoyt PJ Miller. Multispectral imaging with a liquid crystal tunable filter. *Optics in Agriculture, Forestry, and Biological Processing*, (2345):354–365, 1995.

[32] J Kishimoto M Hashimoto. Two-shot type 6-band still image capturing system using commercial digital camera and custom color filter. *International Science and Technology Fourth European Conference on Colourin Graphics*, page 538, 2008.

[33] N Ohyama M Yamaguchi, H Haneishi. Beyond red-green-blue (rgb): spectrum-based color imaging technology. *Journal of the Imaging Science and Technology*, (52), 2008.

[34] GW Fisher D Fishman Y Garini W Niu Wachman DL Farkas, BT Ballou. Optical diagnostics of living cells and biofluids. *SPIE Proceedings, Microscopic and Mesoscopic Spectral Bio-Imaging*, (2678):200–206, 1996.

[35] S Tominaga M Doi, R Ohtsuki. Spectral estimation of skin color with foundation makeup. *Image Analysis, Volume 3540 of Lecture Notes in Computer Science*, pages 95–104, 2005.

[36] Miaohong Shi and Glenn Healey. Using reflectance models for color scanner calibration. *Journal of the Optical Society of America. A, Optics and image science*, 19(4):645–656, 2002.

[37] B Dyas. Robust sensor response characterization. *The International Science andTechnology / SID Eighth Color Imaging Conference*, pages 144–148, 2000.

[38] LT Maloney. *Evaluation of Linear Models of Surface Spectral Reflectance with Small Numbers of Parameters*. Jones and Bartlett Publishers, Inc, USA, 1992.

[39] JP Crettez Y Wu JY Hardeberg H Maitre, F Schmitt. Spectrophotometric image analysis of fine art paintings. *IS&T and SID's 4th Color Imaging Conference: Color Science, Systems and Applications*, pages 50–53, 1996.

[40] L T Maloney. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *Journal of the Optical Society of America. A, Optics and image science*, 3(10):1673–83, October 1986. ISSN 0740-3232. URL http://www.ncbi.nlm.nih.gov/pubmed/3772629.

[41] S Tominaga. Spectral imaging by a multichannel camera. *Journal of Electronic Imaging*, (8):332–341, 1999.

[42] FH Imai. Multi-spectral image acquisition and spectral reconstruction using a trichromatic digital camera system associated with absorption filters. *Technical report, Munsell Color Science Laboratory Technical Report, Rochester*, 1998.

[43] A Lundervold T Taxt. Multispectral analysis of the brain using magnetic resonance imaging. *Medical Imaging IEEE Trans*, (13):470–481, 1994.

[44] UP Wild CU Keller R Gschwind AC Rosselet, W Graff. Persistent spectral hole burning used for spectrally high-resolved imaging of the sun. *Imaging Spectrometry, SPID Proceedings*, (2480):205–212, 1995.

[45] SM Davis PH Swain. *Remote Sensing: The Quantitative Approach*. McGraw-Hill, 1978.

[46] N Tsumura. Appearance reproduction and multispectral imaging. *Color Resources Applied*, (31):270–277, 2006.

[47] Y Ohya T Obi N Ohyama Y Komiya T Wada M Yamaguchi, R Iwama. Natural color reproduction in the television system for telemedicime. *Medical Imaging*, (3031):482–489, 1997.

[48] RS Berns PD Burns. Analysis of multispectral image capture. *Proceedings of the IS&T/SID Fourth Color Imaging Conference: Color Science, Systems, and Applications, Color Imaging Conference*, pages 19–22, 1996.

[49] E Saber AE Ononye, A Vodacek. Automated extraction of fire line parameters from multispectral infrared images. *Remote Sensor Environment*, (108):179–188, 2007.

[50] HH Huang. Acquisition of multispectral images using digital cameras. *Asian Association on Remote Sensing*, 2004.

[51] DW Hillger GP Ellrod, BH Connell. Improved detection of airborne volcanic ash using multispectral infrared satellite data. *J Geophysical Resources*, (108): 4356–4369, 2003.

[52] S Tominaga. Multichannel vision system for estimating surface and illumination functions. *Journal of the Optical Society of America*, (13):2163–2173, 1996.

[53] FW Vorhagen B Hill. Multispectral image pick-up system, 1994.

[54] MH Horman. Temperature analysis from multispectral infrared data. *Applied Optics*, (15):2099–2104, 1976.

[55] K Ohsawa T Uchiyama H Motomura Y Murakami N Ohyama M Yamaguchi, T Teraji. Color image reproduction based on the multispectral and multiprimary imaging: Experimental evaluation. *Color Hardcopy, and Applications VII, SPIE Proceedings*, (4663):15–26, 2002.

[56] Troscianko T Moorehead IR Parraga CA, Brelstaff G. Color and luminance information in natural scenes. *Journal of the Optical Society of America*, (15): 563–569, 1998.

[57] JY Hardeberg DR Connah. Spie proceedings, (spectral recovery using polynomial models). *Color Imaging X: Processing, Hardcopy, and Applications*, (5667):65–75, 2005.

[58] S Westland D Connah, J Hardeberg. Comparison of linear spectral reconstruction methods for multispectral imaging. *CIP04 International Conference on Image Processing*, (3):1497–1500, 2004.

[59] CE Mancill WK Pratt. Spectral estimation techniques for the spectral calibration of a color image scanner. *Applied Optics*, (15):73–75, 1976.

[60] Osorio D. Chiao CC, Cronin TW. Characteristics of reflectance spectra and effects of natural illuminants. *Journal of the Optical Society of America*, (17): 218–224, 2000.

[61] Thomson MGA Connah D, Westland S. Recovering spectral information using digital camera systems. *Color Technology*, (117):309–312, 2001.

[62] T Kanade C Zitnick. A cooperative algorithm for stereo matching and occlusion detection. *Technical report CMU-RI-TR-99-35, Robotics Institute, Pittsburgh*, 1999.

[63] KC Hung YS Chen CS Fuh YP Hung, CS Chen. Multipass hierarchical stereo matching for generation of digitalterrain models from aerial images. *Machine Vision and Applications*, (10):5–6, 1998.

[64] MM Trivedi SB Marapane. Multi-primitive hierarchical (mph) stereo analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (16):227–240, 1994.

[65] M M Hayat, S N Torres, E Armstrong, S C Cain, and B Yasuda. Statistical algorithm for nonuniformity correction in focal-plane arrays. *Applied optics*, 38(5):772–80, February 1999. ISSN 0003-6935. URL http://www.ncbi.nlm.nih.gov/pubmed/18305675.

[66] Russell C Hardie, Majeed M Hayat, Earnest Armstrong, and Brian Yasuda. Scene-based nonuniformity correction using video sequences and registration. *Applied optics*, 39(8):1241–1250, 2000.

[67] Bradley M Ratliff, Majeed M Hayat, and Russell C Hardie. correction in focal-plane arrays. *Journal of the Optical Society of America. A*, 19(9):1737–1747, 2002.

[68] Sergio N Torres, Jorge E Pezoa, and Majeed M Hayat. Scene-based nonuniformity correction for focal plane arrays by the method of the inverse covarianze form. *Applied optics*, 2003.

[69] Bradley M Ratliff, Majeed M Hayat, and J Scott Tyo. Radiometrically accurate scene-based nonuniformity correction for array sensors. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 20(10): 1890–9, October 2003. ISSN 1084-7529. URL http://www.ncbi.nlm.nih.gov/pubmed/14570103.

[70] A Oppenheim. Nonlinear filtering of multiplied and convolved signals. *IEEE Transactions on*, (3), 1968. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1161990.

[71] J Anthony Tyson, T Bell Laboratories, and Murray Hill. Low-light-level charge-coupled device imaging in astronomy. *Journal of the Optical Society of America. A*, 3(12):2131–2138, 1986.

[72] M Thomson and S Westland. by parametric fitting of sensor responses. *Color Research & Application*, 2001.

[73] Marta de Lasarte, Jaume Pujol, Montserrat Arjona, and Meritxell Vilaseca. Optimized algorithm for the spatial nonuniformity correction of an imaging system based on a charge-coupled device color camera. *Applied Optics*, 46 (2):167–74, January 2007. ISSN 0003-6935. URL http://www.ncbi.nlm.nih.gov/pubmed/17268559.

[74] J. Klein. Multispectral imaging and image processing. *Image processing algorithms and systems XII, Society of Imaging Science and Technology*, (90190Q), 2014.

[75] R Schettini P Pellegri, G Novati. *Selection of Training Sets for the Characterisation of Multispectral Imaging Systems*. PICS, 2003.

[76] B Funt K Barnard. Camera characterization for color research. *Color Research and Application*, (27):152–163, 2002.

[77] B. Wandell and J. Farrell. Water into wine: Converting scanner rgb to tristimulus xyz. In *Proc. SPIE*, 1993.

[78] G. Cui Luo, M. R. and B. Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research and Application*, (26):340–350, 2001.

[79] Hugh Fairman. Metameric correction using parameric decomposition. *Color Research and Application*, (5):261–265, 1987.

[80] Roy S. Berns. *Principles of color Technology, 2nd ed.* John Wiley and Sons, 2000.

[81] J Hernandez-Andres J Romero, A Garcia-Beltraan. Linear bases for representation of natural and artificial illuminants. *Journal of the Imaging Science and Technology*, (14):1007–1014, 1997.

[82] R Schettini L Vanneschi S Bianco, F Gasparini. Polynomial modeling and optimization for colorimetric characterization of scanners. *Journal of Electronic Imaging*, (17):4, 2008.

[83] P Gouton A Mansouri, FS Marzani. Neural networks in two cascade algorithms for spectral reflectance reconstruction. *IEEE International Conference on Image Processing*, pages 2053–2056, 2005.

[84] MS Peercy. Linear color representations for full speed spectral rendering (acm, new york). *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 191–198, 1993.