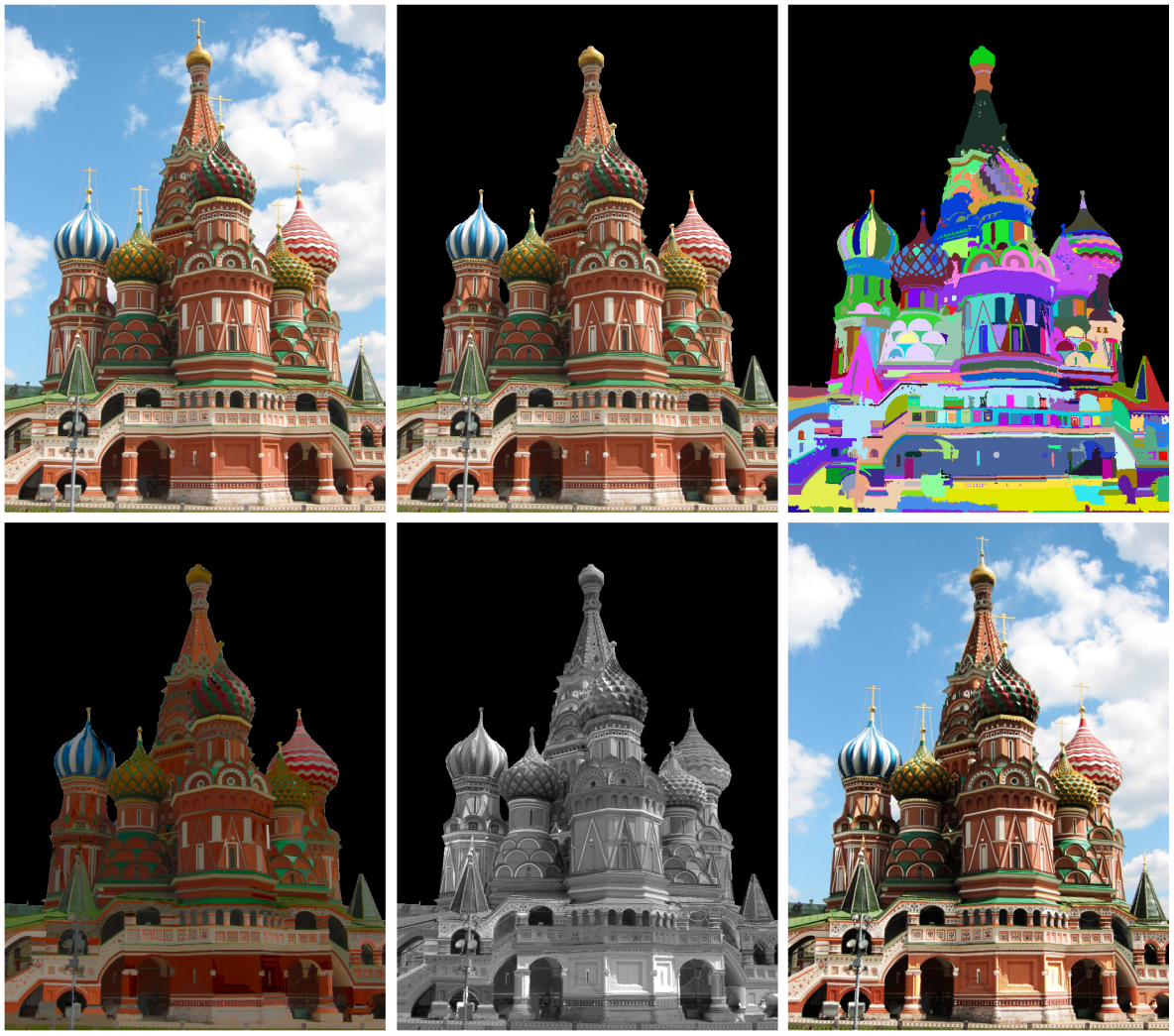
Proyecto Fin de Carrera

Descomposición de imágenes en sus componentes intrínsecas



Autora: Elena Garcés García

Ingeniería Informática. Diciembre 2010

Director:
D. Jorge López-Moreno

Ponente:
Dr. D. Diego Gutiérrez

Universidad de Zaragoza

Centro Politécnico Superior

Departamento de Informática e Ingeniería de Sistemas

Grupo de Informática Gráfica Avanzada

Si quieres conocer el pasado, mira el presente que es su resultado. Si quieres conocer el futuro, mira el presente que es su causa. BUDA

Agradecimientos

Quiero dar las gracias desde aquí a mis padres, Aurora y Cándido, por su apoyo a lo largo de estos años de carrera y por aguantarme en los momentos más difíciles. A mi hermana, Mónica, por no haber dejado de creer en mí. A Carlos, por estar siempre e incondicionalmente, por ayudarme, apoyarme y hacerme ver la luz cuando todo parecía oscuridad.

También quiero agradecer a Jorge y a Diego la oportunidad que me dieron de trabajar en un proyecto tan interesante como ha sido este, por su ayuda, dedicación y paciencia durante todos estos meses. Gracias a toda la gente del grupo por su apoyo y disposición y en especial a Adolfo por sus valiosas aportaciones en la linealización del problema.

Resumen

En múltiples tareas de fotografía computacional, tales como extracción de 3D o edición de materiales e iluminación para una sola imagen, es necesario el conocimiento previo de las luces y los materiales de la escena. En ocasiones podemos disponer de esta información, bien porque nos encontramos en entornos controlados, o bien porque disponemos de la tecnología que captura dichos datos al tomar la imagen. Sin embargo, en la mayoría de los casos no disponemos de tal información y el único modo que tenemos de obtener luces o materiales es a través de una sencilla fotografía. Este proyecto tiene como objetivo resolver este problema que comúnmente se conoce como descomposición de una imagen en sus componentes intrínsecas, y que consiste en obtener, para una única imagen, la parte correspondiente a iluminación (sombreado) y la que corresponde con reflectancia (textura, color).

A lo largo de los años se han desarrollado numerosos métodos para su resolución, sin embargo, el elevado número de incógnitas y la ausencia de información previa de la escena hacen imposible obtener una solución unívoca y óptima. Por ello, para acotar el problema y que sea posible su resolución hemos partido de ciertas asunciones iniciales: suponemos conocido el contorno de los objetos de la imagen y éstos son considerados globalmente convexos.

Nuestro algoritmo, realizado bajo un proyecto en colaboración con Adobe Systems Inc., se basa en encontrar las relaciones de luminosidad entre las distintas regiones de la imagen, para posteriormente normalizarlas, y que de este modo se eliminen las variaciones de luminosidad que sean causadas por la textura de los materiales manteniendo la información de la geometría de la escena. Comparado con otros métodos, nuestro trabajo proporciona una solución precisa y no requiere conocimientos avanzados sobre parámetros del algoritmo ni interacción por parte del usuario, ya que se ejecuta de manera automática. Por este motivo, sirve fácilmente de base para cualquier técnica que requiera de esta descomposición.

Índice general

Capítulo 1

Introducción

Este documento recoge la memoria del Proyecto Fin de Carrera titulado *Descomposición de una imagen en sus componentes intrínsecas*. En este primer capítulo se describen en los aspectos más básicos del proyecto como son la motivación, el contexto en el que se ha desarrollado y los objetivos que se pretendían alcanzar, así como el estado del arte.

1.1. Motivación

En los últimos años está emergiendo un nuevo campo de investigación en el que convergen informática gráfica, visión por computador y fotografía, llamado *fotografía computacional* [RT10]. Su objetivo es superar las inherentes limitaciones de las cámaras digitales convencionales usando técnicas de computación para mejorar la captura de imágenes, su manipulación, y la interacción con los medios visuales. Algunas de estas tareas son la compresión de imágenes de alto rango dinámico para su correcta visualización en los dispositivos (i.e tone mapping), el escalado y el auto-completado de imágenes o la edición de efectos artísticos en imágenes o escenas. Se puede observar el crecimiento de este área por su, cada vez más, significativa aparición en los congresos y revistas más importantes de informática gráfica y visión por computador.

El principal problema que tienen algunas de estas técnicas, en concreto las relacionadas con la edición de imágenes, es la escasez de información que disponen sobre la escena, sobre todo si trabajan con una única fotografía de la misma. En ese caso resulta imposible realizar determinadas tareas que requieran el conocimiento previo de los materiales, la iluminación o la geometría de los objetos, como puede ser extracción de 3D, re-iluminación o colorización de imágenes. Por otro lado, conocer esta información puede ser extremadamente útil en muchas aplicaciones de visión por computador, donde sea necesario desambiguar la iluminación de los materiales para una correcta interpretación de la escena.

El problema de obtener la iluminación y los materiales de una imagen ha sido un reto abierto desde que Barrow y Tenenbaum [BT78] formularan el problema en 1978 con el

(a) Imagen Original (b) Reflectancia (c) Iluminación

Figura 1.1: *Ejemplo de descomposición en imágenes intrínsecas*

nombre de *descomposición en imágenes intrínsecas*. Esta descomposición consiste en separar una imagen en dos componentes: una imagen que representa la reflectancia del objeto (la textura), y otra que representa la iluminación. Un ejemplo de esta descomposición se puede ver en la Figura 1.1.

Desafortunadamente, en una imagen, reflectancia e iluminación están unidas a través de una compleja interacción y, dado que existen múltiples combinaciones, es imposible desambiguarlas. Por ejemplo, ¿cómo sabemos si una zona azul de la imagen pertenece a un material blanco iluminado con luz azul, o a un material azul iluminado con luz blanca? Al cerebro humano no le cuesta realizar esta operación, principalmente, porque dispone de información aprendida de los objetos. Además, tiene la capacidad de eliminar diferencias de color producidas por diferencias en la iluminación, propiedad que se conoce como como constancia del color [EHLM71] y que es una característica deseable en los algoritmos de procesamiento de imágenes.

Este trabajo se encuentra dentro de la línea de investigación en la que colabora Adobe Systems Inc. y que surge de la necesidad de una buena descomposición para integrar en otros algoritmos, por ejemplo los algoritmos de *Shape from Shading* (ver Figura 1.2). El interés de la empresa en el proyecto viene dado por su descontento con las últimas investigaciones que han realizado en el tema, en las que requerían excesiva interacción del usuario [BPD09] (más información en el capítulo 2). Nuestra intención era, por tanto, elaborar un algoritmo que pudiera servir de base para diversas aplicaciones de edición o tratamiento de imágenes sin que supusiera una dificultad añadida al usuario por tener parámetros demasiado complejos y que debía ejecutarse en tiempo interactivo (menos de un minuto).

1.2. Objetivos

Los objetivos que se plantearon inicialmente y que se han cumplido satisfactoriamente son los siguientes:

1. Estudiar las técnicas existentes de descomposición de imágenes intrínsecas.

2. Elaborar un algoritmo propio de obtención de imágenes intrínsecas.

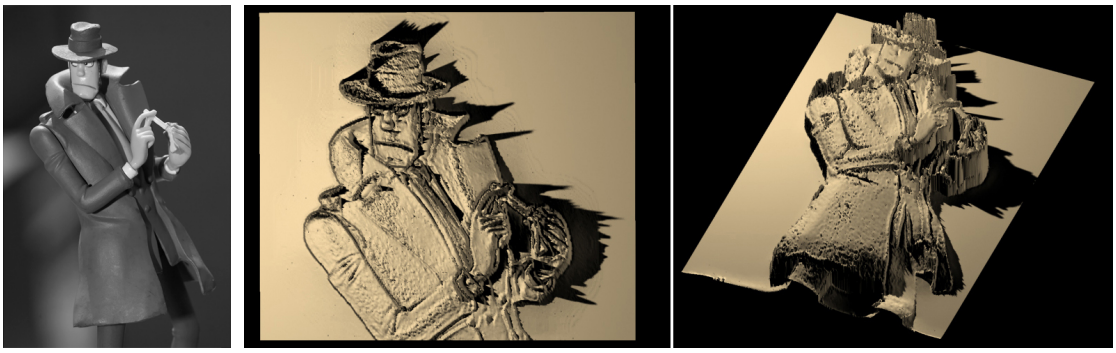
3. Comparar la solución obtenida con los métodos más importantes de descomposición.

1.3. Contexto

El proyecto ha sido desarrollado en los laboratorios del Grupo de Informática Avanzada (GIGA), perteneciente al Departamento de Informática e Ingeniería de Sistemas (DIIS) en el Centro Politécnico Superior (CPS) de la Universidad de Zaragoza. Ha sido dirigido por Jorge López Moreno y supervisado por el Dr. Diego Gutiérrez.

El trabajo está encuadrado dentro de la línea de investigación del GIGA en el campo de la fotografía computacional y edición de imágenes y en colaboración con la empresa Adobe Systems Inc. Asimismo, el proyecto ha sido respaldado parcialmente mediante una beca del programa de iniciación a la investigación del Instituto de Investigación en Ingeniería de Aragón (I3A).

Los resultados de este proyecto serán sometidos al congreso internacional de informática gráfica SIGGRAPH 2011, el cual publica en la revista ACM Transactions on Graphics - JCR, posición 1/86 (Computer Science, Software Engineering), 5-year impact factor 4.997.



(a) (b)

Figura 1.2: *Los algoritmos de* **Shape from Shading** *reconstruyen la forma 3D (b) a partir de la imagen del sombreado (a). Asumen que los objetos son globalmente convexos y parten de la premisa de* dark is deep*, es decir, más brillante más cercano y más oscuro más lejano [LMJH+]*

1.4. Estructura de la memoria

Los siguientes capítulos están estructurados según lo siguiente: en el capítulo 2 se describe desde un punto de vista técnico qué son las imágenes intrínsecas y se muestra un resumen de las técnicas más relevantes de descomposición en imágenes intrínsecas. En el capítulo 3 se describe nuestro algoritmo de descomposición y en el capítulo 4 se encuentran los resultados obtenidos. Finalmente en el capítulo 5 se encuentran las conclusiones y el trabajo futuro.

La memoria cuenta con tres anexos: en el Anexo A, hay una extensión de la sección 3.1 del capítulo 3 donde se explican diversas técnicas de segmentación y se exploran los parámetros del algoritmo escogido. En el Anexo B, se encuentra el estudio de precondicionadores, que es una extensión de la sección 3.2. En el Anexo C, hay otro conjunto de resultados y aplicaciones.

Capítulo 2

Estudio de técnicas de descomposición

En este capítulo se explica qué son las imágenes intrínsecas y como interactúan para formar la imagen real. Posteriormente, se da un resumen de las técnicas más relevantes de descomposición en imágenes intrínsecas.

2.1. Formación de la imagen

La idea de separar reflectancia e iluminación de una imagen fue introducida por Barrow y Tenenbaum [BT78] que denotaron el problema con el nombre de descomposición en *imágenes intrínsecas*. La reflectancia describe cómo un objeto refleja la luz y se conoce también con el nombre de *albedo*. La *iluminación* corresponde con la cantidad de luz que incide en un punto y depende de la orientación de la superficie y las condiciones lumínicas. Aunque es a menudo nombrada como *sombreado* incluye, además de sombras, efectos como la iluminación indirecta. Una formulación simplificada del problema viene dada por lo siguiente: dada un imagen de entrada a la que llamamos $I(x, y)$, la descomposición en sus componentes intrínsecas viene determinada por la ecuación,

$$I(x, y) = S(x, y) \times R(x, y) \tag{2.1}$$

donde $S(x, y)$ es la imagen de la iluminación y $R(x, y)$ es la imagen de la reflectancia (ejemplo gráfico en la Figura 2.1).

Nuestro objetivo es obtener $S(x, y)$ y $R(x, y)$ y, como vemos en la fórmula, puesto que tenemos el doble número de incógnitas que de ecuaciones nos resulta imposible desambiguar para obtener la solución real. En los últimos años, debido a la proliferación de las técnicas de edición de imágenes se han desarrollado numerosas aproximaciones a la solución desde distintas perspectivas. En la siguiente sección se da un resumen de las más relevantes.

(a) Imagen original (b) Iluminación (c) Reflectancia

Figura 2.1: ***Descomposición***. *La imagen (a) está formada por sus componentes intrínsecas (b) y (c)*

2.2. Estado del Arte

Weiss [Wei01] propone un método para encontrar las imágenes intrínsecas tomando como entrada una larga secuencia de imágenes de la misma escena en las que la reflectancia permanece constante mientras que la iluminación se modifica. Esta aproximación fue extendida por Liu et al. [LWQ+08] a cualquier secuencia no controlada de imágenes de la escena para dar color a fotografías en blanco y negro. Sin embargo, estas técnicas requieren demasiadas imágenes de entrada para resultar prácticas.

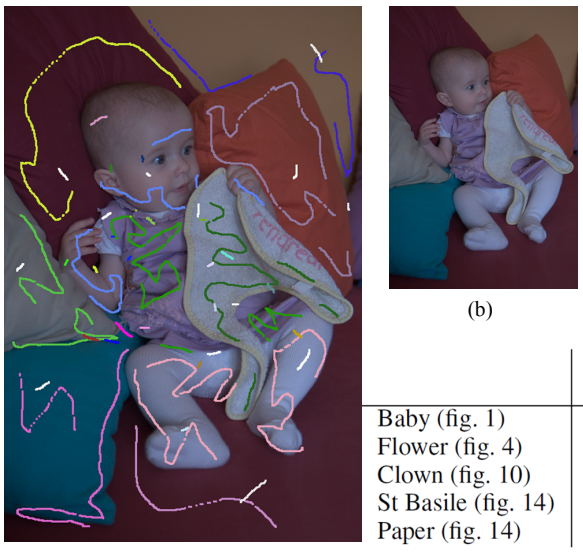
Debido a la falta de variables que acoten el problema, obtener las descomposición a partir de una única imagen no puede ser resuelto sin algún conocimiento a priori de la escena. Por ello, basándose en la teoría Retinex [EHLM71], Horn [Hor86] en su trabajo, asume que mientras que la reflectancia se mantiene constante por segmentos, la iluminación varía suavemente. Esta heurística permite obtener la reflectancia de una imagen umbralizando los gradientes pequeños de la imagen ya que se consideran parte de la iluminación. Tappen et al. [TFA05] cuentan con unos clasificadores entrenados en las derivadas de la imagen para distinguir la reflectancia de los gradientes de iluminación. A pesar de estas heurísticas y clasificadores muchas configuraciones de reflectancia e iluminación continúan siendo difíciles de desambiguar (ver Figura 2.2). Shen et al. [STL08] proponen enriquecer estas aproximaciones con restricciones globales en la textura y partiendo de un algoritmo de *Retinex*, imponen que pixels que comparten textura tengan la misma reflectancia.

Los últimos avances publicados por Adobe Systems Inc. en el tema de obtención de imágenes intrínsecas están recogidos en el artículo de Bousseau et al. [BPD09]. Los cuales, obtienen muy buenos resultados asumiendo que la reflectancia presenta variaciones de bajo rango a nivel local. Sin embargo, requieren excesiva ayuda de un usuario que esté familiarizado con la técnica, ya que el uso de sus pinceles no es intuitivo (ver Figura 2.3).

Existe un extenso trabajo sobre la eliminación de sombras, tanto de manera automática [GDFL04, FHLD06] como basado en interacción con el usuario [MTC07, WTBS07]. La idea común de estos métodos consiste en identificar los pixels de sombra a través de la detección de bordes o de la segmentación de la imagen. Una vez que las sombras son

(a) Imagen original (b) Iluminación (c) Reflectancia

Figura 2.2: ***Error en la descomposición [TFA05]***. *La variación blanco sobre negro del ojo de la figura se puede interpretar erróneamente como sombreado, siendo en realidad parte de la textura.*



(a)

(b)

(c)

(d)

	Res.	User strokes	Matrix (s)	Solving (s)
Baby (fig. 1)	533 × 800	58	20.64	11.93
Flower (fig. 4)	750 × 500	15	18.21	7.87
Clown (fig. 10)	486 × 800	33	18.83	9.53
St Basile (fig. 14)	800 × 600	81	23.29	14.56
Paper (fig. 14)	750 × 500	36	18.17	9.2

(e)

Figura 2.3: ***Interacción del usuario de Bousseau et al. [BPD09]***. *En (a) vemos los trazos necesarios para obtener la descomposición de la imagen (b), en reflectancia (c) e iluminación (d). En la tabla (e) se muestra el número total de trazos que han necesitado algunas imágenes para descomponerse según su algoritmo.*

detectadas pueden ser eliminadas aplicando corrección de color o filtros de gradiente. Sin embargo, estos métodos se centran en capturar sombras proyectadas las cuales se caracterizan por tener fronteras bien diferenciadas, mientras que nosotros tenemos como objetivo eliminar sombreados suaves donde los límites entre luz y oscuridad no pueden ser delimitados. Hay que destacar que aunque la aproximación de Finalyson et al. [GDFL04] tiene como objetivo estimar una imagen sin iluminación, esta imagen es en escala de grises y no representa la verdadera reflectancia.

La obtención de imágenes intrínsecas está estrechamente relacionado con otros tipos de descomposición. Las descomposiciones automáticas en múltiples escalas que capturan

diferentes niveles de detalle [SSD09, FAR07, FFLS08] pueden producir imágenes muy similares a las que se pretende obtener en las imágenes intrínsecas. Algunos de estos algoritmos como el Filtro Bilateral [CPD07] suelen ser utilizados como base para obtener la iluminación en algoritmos de *Shape from Shading* (obtención de 3D). El algoritmo de descomposición muli escala de Subr et al. [SSD09], captura en los niveles de menor detalle, características globales de la iluminación que podrían servir de ayuda en la descomposición.

Capítulo 3

Descomposición en reflectancia e iluminación

Nuestro algoritmo de descomposición es un proceso automático (no requiere intervención del usuario) que toma como entrada la imagen original y una máscara en blanco y negro que define el objeto que queremos descomponer. Consta de dos fases principales fácilmente modularizables. Una visión general de nuestro sistema se puede ver en la Figura 3.1:

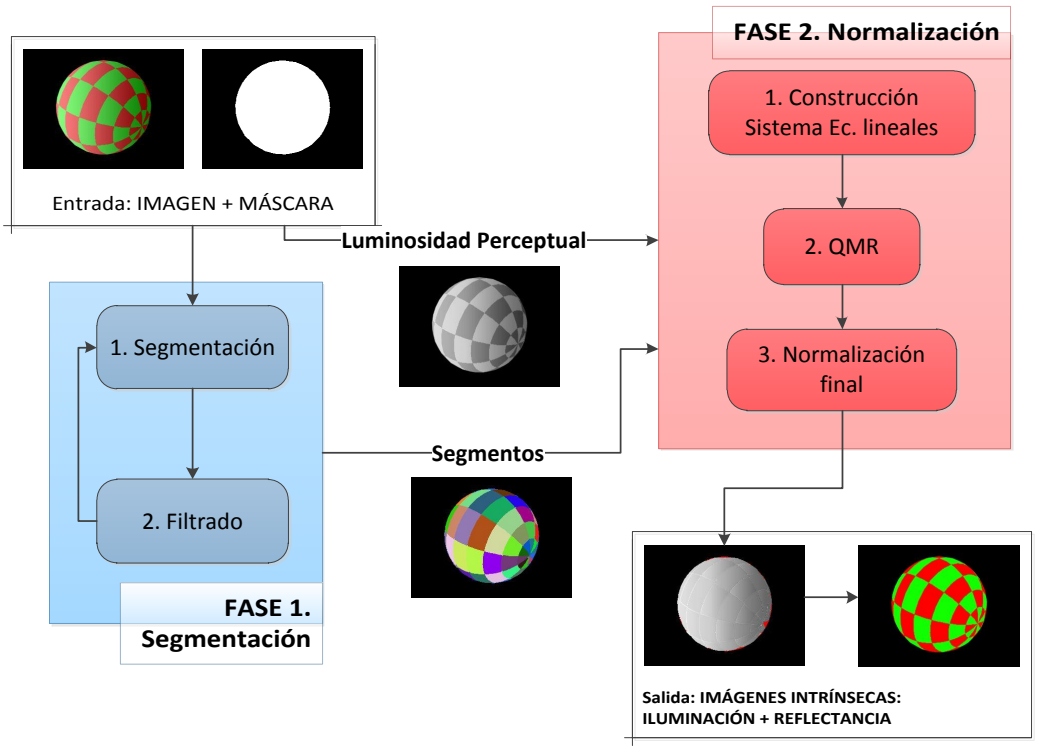


Figura 3.1: *Visión general del sistema implementado.*

- En la primera fase dividimos la imagen en pequeños fragmentos de color o albedo constante. Para ello, utilizamos un método de segmentación basado en grafo [FH04b] modificado para trabajar sobre espacio de color Lab (más detalles sobre la segmentación en la sección 3.1). Estos segmentos (o clusters de pixels) servirán como entrada a la segunda fase del algoritmo junto con la imagen que representa la luminosidad perceptual de la imagen de entrada.
- En la segunda fase (sección 3.2) construimos un sistema de ecuaciones estableciendo relaciones locales de luminosidad entre pares de clusters vecinos de la imagen. Como resultado, obtenemos unos ratios de luminancia para cada cluster que nos permiten llegar a la imagen final de la iluminación.
- A partir de la imagen normalizada de la iluminación, calculamos la imagen correspondiente de la reflectancia. Este par de imágenes constituyen las imágenes intrínsecas buscadas.

3.1. Fase 1: Segmentación

Segmentar una imagen consiste en dividirla en las diferentes partes que la integran. Esto puede parecer sencillo, sin embargo, en la práctica, decidir qué es una buena segmentación es algo muy subjetivo y depende mucho de la finalidad de la aplicación. Por este motivo, elegir un buen algoritmo de segmentación que se ajustara a nuestras necesidades no fue fácil. Evaluamos varias de las técnicas de segmentación existentes partiendo de unos requisitos iniciales: la segmentación no debía ser supervisada (no conocemos a priori el número de clusters ni sus características), se debía usar información de color y/o de textura y el algoritmo debía ser eficiente.

Teniendo esto en mente, estudiamos algoritmos basados en Campos Aleatorios de Markov (MRFs), métodos de agrupamiento y métodos basados en grafos. El estudio completo de los distintos métodos de segmentación se encuentra en el Anexo A. Después de analizar los distintos métodos decidimos escoger el algoritmo de segmentación basado en grafo de Pedro F. Felzenszwalb [FH04b] con algunas modificaciones que se detallan a continuación.

3.1.1. Segmentación basada en grafo

El algoritmo parte de un grafo no dirigido $G = (V, E)$ formado por un conjunto de vértices $v_i \in V$, que se corresponden con los pixels a segmentar de la imagen, y un conjunto de aristas $(v_i, v_j) \in E$ que constituyen pares de vértices vecinos. Cada arista tiene tiene un peso $w((v_i, v_j))$, que representa la similitud entre los dos pixels conectados por esa arista. En el artículo original se proponen dos estructuras de grafo distintas, una basada en una malla (grafo *GRID*), donde cada pixel está conectado con sus 8-vecinos más próximos por posición y otra basada en el método de vecino más próximo (grafo *KNN*) donde se

realiza un mapeo de cada pixel en un espacio nuevo de características y se puede definir libremente el número de vecinos a usar.

En el caso de usar un grafo *GRID*, la función que define la similitud entre los dos pixels conectados por una arista, viene dada por su diferencia de color. Como propone el autor, usamos la distancia Euclídea L_2,

$$w((v_i, v_j)) = \|C(v_i) - C(v_j)\| = \sqrt{\sum_{t=1}^{N} |C(v_i)_t - C(v_j)_t|} \tag{3.1}$$

donde $C(v)$ es el vector de color del vértice v, siendo $C(v) = \{r, g, b\}$ en espacio de color *RGB* y $C(v) = \{a, b\}$ en espacio de color *Lab* (más detalles sobre el modelo de color usado en la sección 3.1.2).

En el otro caso, usando grafo *KNN* se realiza un mapeo de cada vértice en el espacio $\{x, y, C(x, y)\}$, donde (x, y) es la localización del vértice en la imagen y $C(x, y)$ representa el color del punto que dependerá del modelo usado. Del mismo modo que con el grafo *GRID* usamos la distancia Euclídea L_2 para definir los pesos de las aristas, pero ahora, también tiene influencia la posición del pixel en la imagen. La ventaja de usar KNN frente a Grid, es que en el primero, el poder seleccionar un número variable de vecinos y capturar en la función de similitud la posición junto con el color, permite que las conexiones de los píxeles sean más flexibles al poderse conectar regiones físicamente separadas. En cambio, usando grafo Grid tenemos una estructura rígida en la que solo podemos realizar conexiones locales.

Para realizar la segmentación de la imagen, el algoritmo trata de localizar los límites entre regiones comparando dos cantidades: una basada en las diferencias de intensidades entre regiones limítrofes y la otra basada en las diferencias de intensidades dentro de cada región. Intuitivamente, la diferencia de intensidad entre dos regiones es perceptualmente importante si es más grande que la diferencia dentro de al menos una de las dos regiones. En el proceso de segmentación, los pixels se distribuyen formando distintas regiones, que se van modificando hasta que el sistema queda equilibrado y la cohesión interna entre los pixels de cada región es suficientemente fuerte. La explicación detallada de cómo funciona el algoritmo de segmentación se encuentra en el Anexo A.

3.1.2. La influencia del modelo de color: RGB vs Lab

El artículo original realiza la segmentación de la imagen usando el espacio de color RGB. A pesar de que los resultados son buenos, no sirven totalmente para nuestros propósitos. Al trabajar sobre el espacio de color RGB, tratan del mismo modo todos los pixels de la imagen y no tienen en cuenta que en una región pueda haber variaciones de luminosidad producidas por sombreado. Por ello, puesto que nosotros buscamos regiones

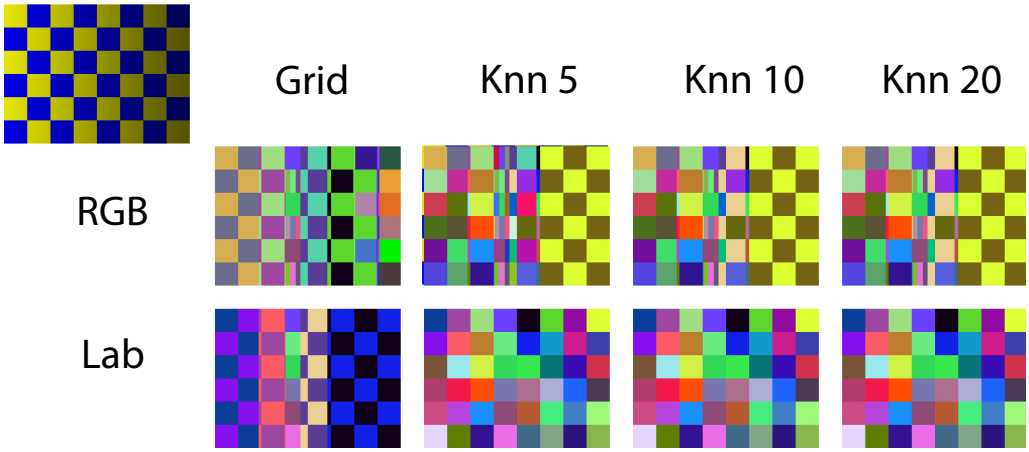


Figura 3.2: ***Comparación de segmentación RGB vs Lab.*** *Para cualquier tipo de grafo, la mejor segmentación se obtiene con espacio de color Lab*

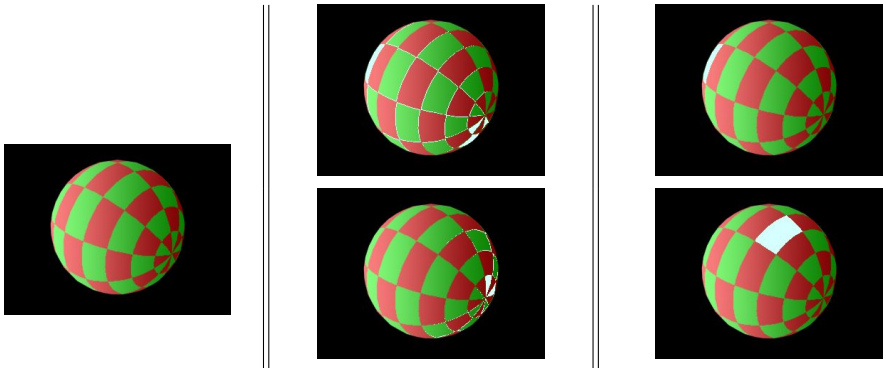
de albedo (reflectancia) constante y basandonos en el estudio de Funt et al. [FDB91] que dice que variaciones de reflectancia alteran la cromaticidad mientras que variaciones de sombreado la mantienen constante, utilizamos el modelo de color Lab [1] . Este modelo, por la forma en que está definido, nos permite abstraernos de estas variaciones lumínicas y trabajar directamente con la crominancia.

En concreto, usamos únicamente los canales cromáticos a y b con lo que conseguimos que una región con un albedo constante, pero sometida a un foco de luz y por ello, con un gradiente de intensidad, sea segmentada como un único cluster y no como varios, como ocurre en el ejemplo de la Figura 3.2 para el caso de RGB.

3.1.3. Filtrado y mejora de la segmentación

Para mejorar los resultados de la segmentación se somete la imagen segmentada inicial a un proceso iterativo de filtrado y re-segmentación de manera que los clusters resultantes tengan la máxima coherencia interna posible. Con este filtrado conseguimos que casos de segmentación que pueden ser un problema para la posterior fase del algoritmo, sean mejorados. En la figura 3.3 vemos un ejemplo de clusters que pueden producir grandes errores en el algoritmo.

[1] El modelo Lab consta de tres dimensiones, la primera dimensión L, representa la luminosidad del color. Las dimensions a y b representan la cromaticidad del color.



(a) Imagen Original | *(b) Mala segmentación* | *(c) Buena segmentación*

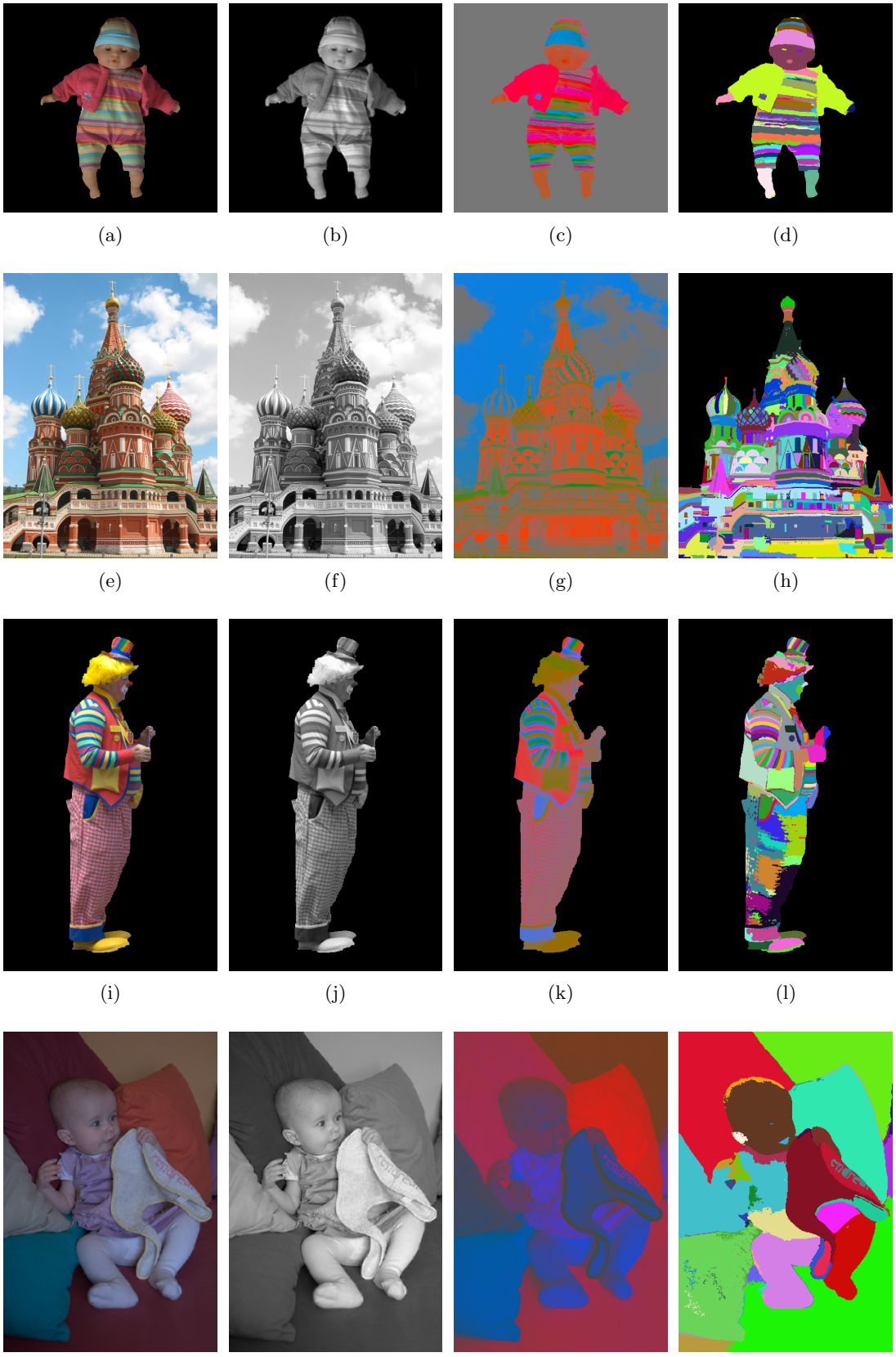
Figura 3.3: ***Ejemplos de segmentación.*** *Los pixels blancos representan un cluster. En (b) los clusters seleccionados abarcan zonas de la imagen muy distantes entre sí y con valores de luminancia muy dispares.*

3.1.4. Resultados segmentación

Para evaluar el algoritmo, realizamos varias pruebas utilizando grafos *GRID* y *KNN* (variando el número de vecinos de 5 a 30). Asimismo, también probamos utilizando el modelo de color Lab y RGB tanto con imágenes artificiales como con imágenes reales. Las pruebas que realizamos nos permiten concluir que los mejores resultados se obtienen usando un modelo de color Lab, que permite una mejor clasificación de clusters por albedo, y grafo KNN, puesto que captura mejor las relaciones de similitud entre píxeles próximos en la imagen. En el Anexo A hay una exhaustiva evaluación del algoritmo original de segmentación donde, además de evaluar los tipos de grafos, se analizan los parámetros de entrada que requiere la segmentación. En nuestro caso, hemos podido fijar los valores de dichos parámetros y mantener el método exento de la intervención del usuario. Vemos algunos ejemplos de segmentación en la Figura 3.4.

3.2. Fase 2: Normalización

Nuestro objetivo es normalizar la imagen de luminancia inicial eliminando aquellas variaciones de luminosidad producidas por textura pero manteniendo las variaciones producidas por la geometría del objeto (sombras). Para ello, comenzamos calculando la imagen de la luminosidad inicial utilizando los valores RGB de cada pixel con la siguiente ecuación $L(x,y) = 0{,}212R(x,y) + 0{,}715G(x,y) + 0{,}072B(x,y)$ [I.T90]. La imagen obtenida representa la luminosidad de la imagen de manera similar a cómo la percibe el ojo humano, sin embargo, no representa la luminosidad real (iluminación) del objeto, ya que contiene toda la información relacionada con la textura de los materiales. Por este motivo, basándonos en la premisa de que el objeto es globalmente convexo y que la iluminación varía suavemente en la superficie del objeto [Hor86], en nuestro algoritmo asumimos que la luminosidad se mantiene constante en las fronteras entre clusters y planteamos un sistema de ecuaciones



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (ñ) (o)

Figura 3.4: *Ejemplos de segmentación. De izquierda a derecha las columnas representan: la imagen original (a)(e)(i)(m), el canal de luminancia perceptual L (b)(f)(j)(n), los canales cromáticos ab (c)(g)(k)(ñ) y la segmentación obtenida (d)(h)(l)(o).*

para encontrar los factores o ratios que relacionan las luminosidades entre los pares de clusters vecinos.

3.2.1. Linealización del problema

Dado un conjunto C de clusters, queremos encontrar unos factores F_c por los que multiplicar cada cluster de la imagen de luminosidad inicial L para obtener la imagen de luminosidad (o luminancia) normalizada L_n,

$$L_n(x, y) = F_c\, L(x, y) \tag{3.2}$$

para $c \in C$ y $(x, y) \in c$.

Puesto que queremos igualar el valor de la luminosidad en las fronteras de los clusters, tenemos una ecuación para cada par de clusters vecinos donde expresamos esta igualdad,

$$F_{c_i}\, L_m(c_i)_{c_j} - F_{c_j}\, L_m(c_j)_{c_i} = 0 \tag{3.3}$$

donde $L_m(c_i)_{c_j}$ representa la luminosidad media de los pixels del cluster c_i que se encuentran en la frontera con el cluster c_j y, F_{c_i} y F_{c_j} representan los factores por los que hay que multiplicar cada cluster para que se igualen sus luminancias.

El conjunto de ecuaciones formado por cada pareja de clusters vecinos nos da un sistema lineal de M ecuaciones y N incógnitas, siendo M el número de pares de clusters adyacentes y N el número total de clusters de la imagen. Como se puede observar, siendo un sistema con más ecuaciones que incógnitas la solución que obtendríamos resolviéndolo es la trivial: $F_{c_1} = F_{c_2} = F_{c_N} = 0$. Por este motivo, se añade una nueva ecuación que conserva la luminosidad total de la imagen y evita la solución trivial,

$$\sum_{i=1}^{N} F_{c_i}\, L_{Me}(c_i) = \sum_{i=1}^{N} L_{Me}(c_i) \tag{3.4}$$

donde L_{Me} es la luminosidad media total de cada cluster. Esta ecuación la denominamos ecuación de conservación de la energía, ya que obliga al sistema a mantener equilibrados los valores de luminancia total de la imagen.

Las ecuaciones 3.3 y 3.4 forman el sistema $AX = B$ para N clusters y M pares de clusters vecinos. Cada fila a_i de A viene definida por,

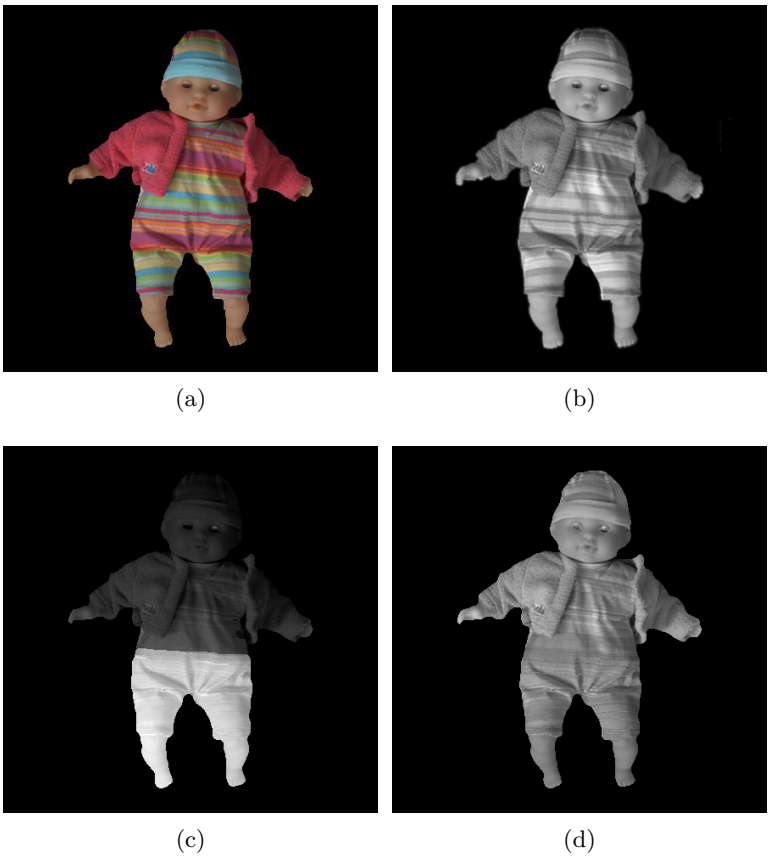
$$\forall i \in 1..M, a_i = \begin{cases} \exists k, l \in 1..N \ni a_{ik} = L_m(c_k)_{c_l},\ a_{il} = -L_m(c_l)_{c_k} \\ \text{con } k < l \wedge c_k \text{ es adyacente a } c_l \\ \\ a_{ih} = 0, \forall h \in 1..N \ni h \neq k \wedge h \neq l \end{cases} \tag{3.5}$$

$$i = M + 1, \forall j \in 1..N, a_{ij} = L_{Me}(c_j)$$

Y X y B definidos por,

$$X_{N\times 1} = \begin{pmatrix} F_{c_1} \\ F_{c_2} \\ \vdots \\ F_{c_N} \end{pmatrix} \qquad B_{(M+1)\times 1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \sum_{i=1}^{N} L_{Me}(c_i) \end{pmatrix} \tag{3.6}$$

Al resolver el sistema equivalente $(A^T A)X = (A^T B)$ usando el método Quasi-Minimal residual (QMR) [BR94], observamos que la solución obtenida no es la esperada. Como se puede ver en la Figura 3.5c, la imagen se polariza y la distribución de la luminosidad no queda uniforme. Esto se debe, principalmente, a que la ecuación 3.4 mantiene la energía global del sistema pero no impide que la distribución de la misma sea desigual. Por otra parte, el hecho de que el vector B esté compuesto valores nulos hace que la resolución del sistema tienda a obtener la solución trivial. Para solucionar estos problemas hemos establecido una similitud con los problemas de termodinámica relacionados con la mecánica de fluidos como se muestra a continuación.



(a) (b)

(c) (d)

Figura 3.5: ***Normalización de luminancias.*** *(a) es la imagen de entrada original, (b) es la imagen de la luminosidad perceptual, (c) es la luminosidad final con el sistema de energías (Ecuación 3.5 y 3.6) y (d) es la luminosidad con el sistema de flujos (Ecuaciones 3.9 y 3.10)*

Buscando el equilibrio lumínico

Nuestro sistema parte de un estado inicial inestable: existe un desequilibrio lumínico entre los clusters, es decir, las transiciones entre los mismos no son uniformes. Y tiene como objetivo buscar el estado de equilibrio, una imagen suave sin saltos bruscos de luminancias. Para ello el sistema va a tratar de equilibrarse realizando transferencias de luminosidad entre sus regiones adyacentes. Esta transferencia lumínica que entra o sale de cada cluster la denominamos *flujos* φ. Para construir el nuevo sistema de flujos partimos del sistema anterior y transformamos las ecuaciones tomando logaritmos, teniendo ahora sumas en lugar de productos, lo cual, como veremos, nos da mayor flexibilidad para modelar el sistema. La ecuación 3.3 queda del siguiente modo:

$$\ln(L_m(c_i)_{c_j}) - \ln(L_m(c_j)_{c_i}) = \varphi_j - \varphi_i \tag{3.7}$$

donde $\varphi_i = \ln(F_{c_i})$ y $\varphi_j = \ln(F_{c_j})$.

La ecuación 3.4 de conservación de la energía del sistema anterior se transforma en una ecuación de conservación de flujos de la siguiente manera,

$$\sum_{i=1}^{N} \varphi_i = 0 \tag{3.8}$$

Con este planteamiento reconstruimos el sistema de ecuaciones anterior $AX = B$. Del mismo modo que antes, A es una matriz de N columnas y $M+1$ filas siendo M el número de pares de clusters adyacentes y N el número de clusters de la imagen. En este caso, cada fila a_i de A viene definida por,

$$\forall i \in 1..M, a_i = \begin{cases} \exists k, l \in 1..N \ni a_{ik} = 1,\ a_{il} = -1 \\ \text{con } k < l \wedge c_k \text{ es adyacente a } c_l \\ \\ \forall h \in 1..N,\ a_{ih} = 0,\ \text{si } h \neq k \wedge h \neq l \end{cases} \tag{3.9}$$

$$i = M+1, \forall j \in 1..N, \quad a_{ij} = 1$$

Y X y B definidos por,

$$X^T_{N\times 1} = \begin{pmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_N \end{pmatrix} \tag{3.10}$$

$$\begin{aligned} &B^T_{(M+1)\times 1} = \begin{pmatrix} b_1 & \dots & b_M & 0 \end{pmatrix} \text{ donde,} \\ &\forall i \in 1..M, \quad \exists k, l \in 1..N \ni b_i = \ln(L_{Me}(c_l)) - \ln(L_{Me}(c_k)) \\ &\text{con } k < l \wedge c_k \text{ es adyacente a } c_l \end{aligned} \tag{3.11}$$

Ahora el vector B está formado por valores distintos de cero, lo cual hace que la resolución del sistema converja a una solución más óptima. Los resultados, como podemos comprobar si observamos la imagen de la Figura 3.5d, son considerablemente mejores que en el caso anterior. No obstante, a pesar de que el algoritmo funciona bien con imágenes sencillas, si lo aplicamos a una imagen con múltiples clusters y numerosas interacciones entre ellos comprobamos que la solución no es del todo buena: todavía persiste el desequilibrio lumínico global de la imagen a pesar de que localmente la distribución es correcta (ver imagen 3.6c). Para impedir este caso, añadimos una ecuación por cada cluster que obliga a que su luminosidad final no se desvíe de la media del sistema:

$$\forall j \in 1..N, \quad \frac{1}{N}\sum_{i=1}^{N}\varphi_i + \ln(L_{Me}(c_i)) = \varphi_j + \ln(L_{Me}(c_j)) \tag{3.12}$$

Esta última restricción la podemos fijar en el sistema actual de flujos, sin embargo, no era posible imponerla en el sistema anterior donde las luminancias eran factores. En ese caso, tendríamos que haber igualado la luminancia de cada cluster a la media geométrica de la imagen con la ecuación,

$$\forall j \in 1..N, \quad (\prod_{i=1}^{n} F_{c_i} L_{Me}(c_i))^{1/n} = F_{c_j} L_{Me}(c_j) \tag{3.13}$$

Finalmente, añadiendo la ecuación 3.12 al sistema de flujos que teníamos en 3.9 y 3.10 obtenemos el sistema de flujos *equilibrado*. La matriz A tendrá $M + N + 1$ filas y N columnas, donde $M + 1$ filas vienen definidas por la ecuación 3.9 y las N filas siguientes por:

$$\forall j \in 1..N, \quad a_{M+1+j} = \begin{cases} \exists k \in 1..N \ni a_{M+1+j,k} = -1, \text{ si } k = j \\ \\ \forall h \in 1..N, \quad a_{M+1+j,h} = \frac{1}{N}, \text{ si } h \neq j \end{cases} \tag{3.14}$$

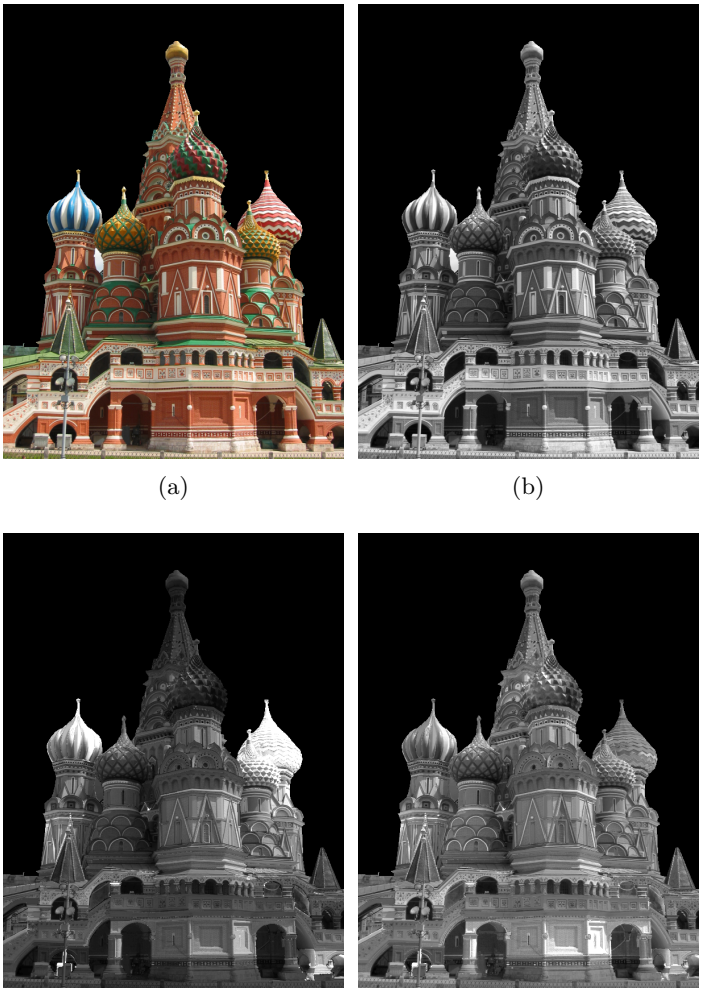
El vector X se mantiene constante y el vector B queda de la siguiente manera:

$$\begin{aligned} &B^T_{(M+N+1)\times 1} = \begin{pmatrix} b_1 & \dots & b_M & 0 & b'_1 & \dots & b'_N \end{pmatrix} \text{ donde,} \\ &\forall i \in 1..N, \quad b'_i = \ln(L_{Me}(c_i)) - \tfrac{1}{N}\textstyle\sum_{j=1}^{N}\ln(L_{Me}(c_j)) \end{aligned} \tag{3.15}$$

Podemos ver cómo mejora notablemente la solución en la imagen de la Figura 3.6d.

3.2.2. Resolución del sistema

Como ya se ha mencionado anteriormente resolvemos el sistema con el método numérico Quasi-Minimal residual (QMR) [BR94] que es muy rápido para resolver este tipo de sistemas lineales [NS94]. La resolución del sistema nos da los valores φ_c buscados. Realizando el cambio de variable $F_c = \exp^{\varphi_c}$ obtenemos los ratios por los que multiplicar los clusters de la imagen y de este modo obtener la imagen buscada de la luminancia.



(a) (b)

(c) (d)

Figura 3.6: ***Normalización de luminancias.*** *(a) es la imagen de entrada original, (b) es la imagen de la luminosidad perceptual, (c) es la luminosidad final con el sistema de flujos (Ecuaciones 3.9 y 3.10) y (d) es la luminosidad final con el sistema de flujos equilibrado(Ecuación 3.12)*

Otra posibilidad que nos planteamos, en lugar linealizar el problema, fue construir una función de minimización de energía que resolveríamos con métodos de optimización. Sin embargo, a pesar de que en ese caso teníamos garantía de éxito, era a costa de un aumento muy considerable en el tiempo de cálculo, lo cual iba en contra de nuestro objetivo de ejecución en tiempo interactivo. QMR, por su parte, es excepcionalmente rápido aún cuando el sistema está compuesto por cientos de ecuaciones. Asimismo, la idea de linealizar el problema nos da la posibilidad de perfeccionar el sistema con precondicionadores haciendo que mejore sustancialmente el tiempo de cálculo y la solución. En el Anexo B hay un estudio completo sobre el uso de precondicionadores en las ecuaciones.

Capítulo 4

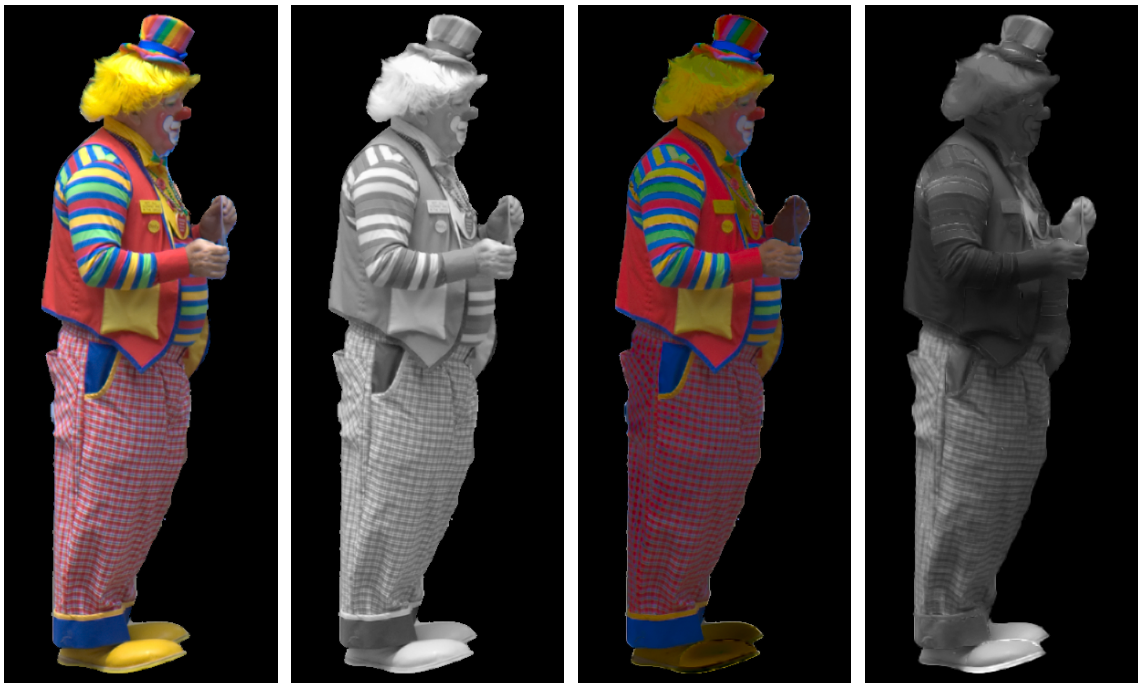
Resultados

Hemos probado nuestro algoritmo de descomposición en un conjunto variado de imágenes. En las Figuras 4.2 , 4.1 y 4.3 vemos la descomposición de las imágenes en sus componentes intrínsecas con nuestro algoritmo. En las Figuras 4.4, 4.5 y 4.6 comparamos nuestros resultados con los obtenidos por las investigaciones más relevantes hasta la fecha en algoritmos *automáticos*: el método de Tappen et al. [TFA05] y el método de Shen [STL08]. Y en algoritmos que requieren de más información: el método de Weiss [Wei01] que necesita varias imágenes de la escena y el algoritmo de Bousseau [BPD09], el cual requiere interacción del usuario con pinceladas. Los resultados demuestran que nuestro método supera en la mayoría de las ocasiones al resto, y como mínimo, los resultados son equiparables (más resultados en el Anexo C). Asimismo disponemos de un conjunto de imágenes que demuestran que nuestra investigación está al nivel, y supera en varios aspectos las últimas investigaciones de Adobe. No obstante, por motivos de propiedad intelectual no nos está permitido mostrarlos en este documento pero serán mostrados en la presentación.

Si observamos la Figura 4.1d comprobamos como hemos capturado correctamente a nivel global la iluminación de la escena. Hemos podido normalizar muy bien las franjas de color de la camisa convirtiéndola en una superficie casi homogénea sin variaciones debidas a textura, sin embargo, no hemos sido capaces de abstraernos de los patrones de textura de algo rango (*cuadritos*) del pantalón del payaso. Este tipo de problemas esperamos solucionarlos analizando la imagen en distintos niveles de detalle, lo cual nos permitirá capturar y tratar estos patrones de forma independiente al resto de la imagen. Del mismo modo, analizando las imágenes de la reflectancia de 4.2b y 4.3b observamos como estas son prácticamente planas, sin apenas variaciones de iluminación ni sombreados. Estas variaciones, por el contrario, se encuentra en la imagen de la iluminación. Podemos comprobar como, por ejemplo, en la imagen del bebé 4.3c han desaparecido completamente las letras del babero, lo cual es una muestra del buen funcionamiento de nuestro algoritmo.

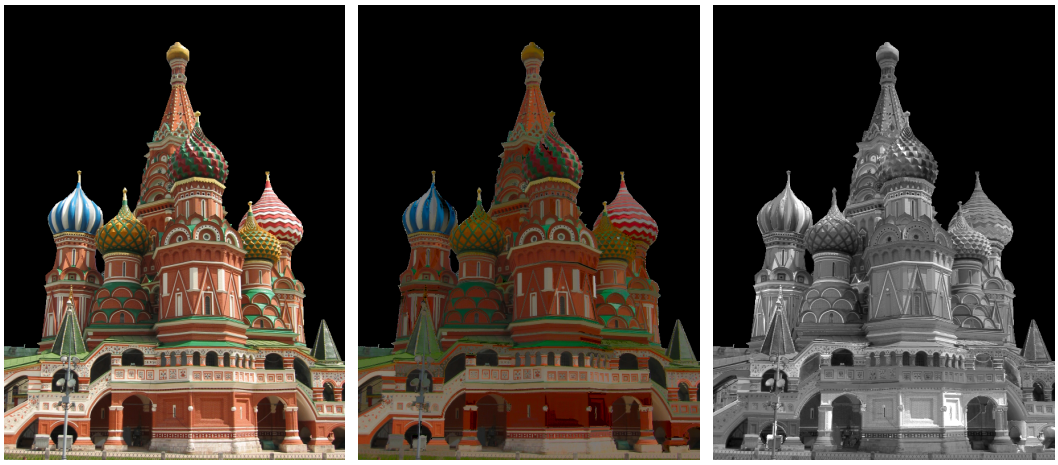
En las Figuras 4.4, 4.5 y 4.6 comparamos nuestro método con los más representativos. En el caso de la Figura 4.4, observamos como ninguno de los métodos ha sido capaz de

eliminar las pintadas del Jaguar de manera correcta. Nosotros, en cambio gracias a nuestro efectivo algoritmo de segmentación somos capaces de detectarlas y eliminarlas. En 4.6, nuestros resultados se parecen mucho a los de Shen et al. [STL08] y Bousseau [BPD09], aunque estos últimos requieren mucha interacción como vemos en 4.6f.



(a) (b) (c) (d)

Figura 4.1: ***Imágenes intrínsecas obtenidas por nuestro algoritmo***. *(a) Imagen Original. (b) Luminancia perceptual. (c) Reflectancia. (d) Iluminación.*

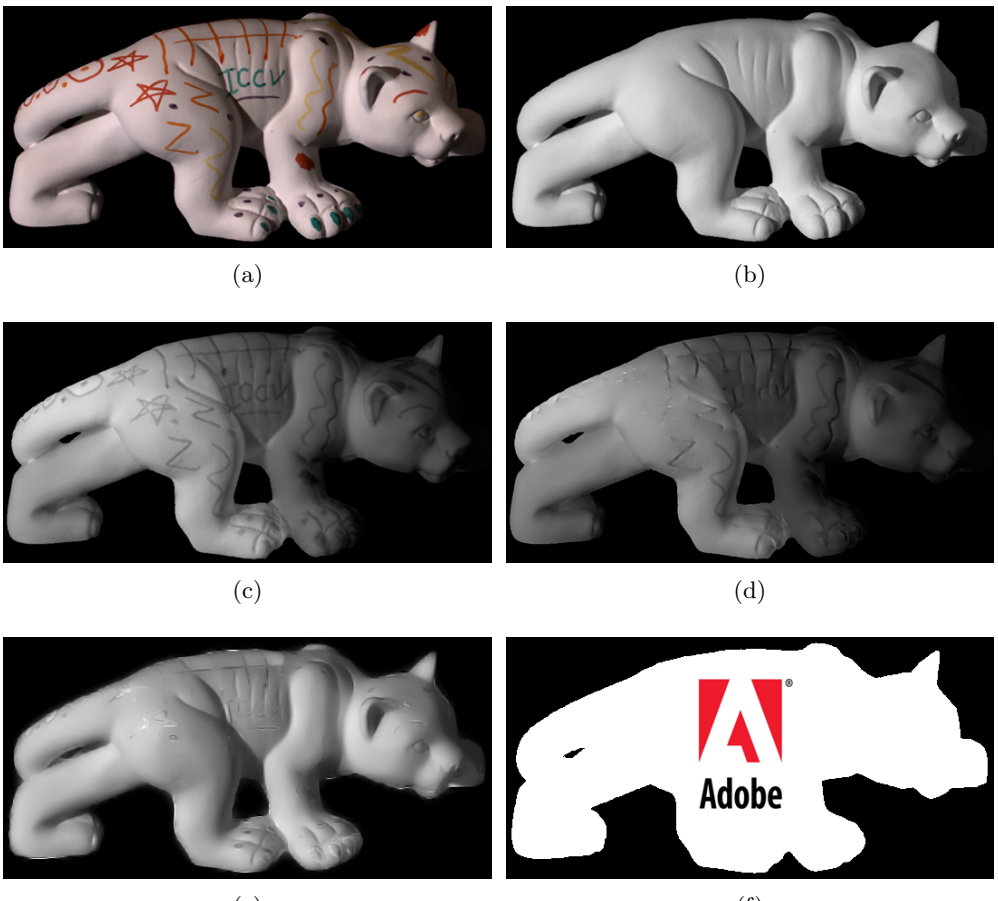


(a) (b) (c)

Figura 4.2: ***Imágenes intrínsecas obtenidas por nuestro algoritmo***. *(a) Imagen Original. (b) Reflectancia. (c) Iluminación. Imagen original por Captain Chaos, flickr.com*

(a) (b) (c)

Figura 4.3: ***Imágenes intrínsecas obtenidas por nuestro algoritmo***. *(a) Imagen Original. (b) Reflectancia. (d) Iluminación.*



(a) (b)

(c) (d)

(e) (f)

Figura 4.4: ***Comparación de la componente de iluminación con otros métodos***. *(a) Representa la imagen real (b) Representa la solución correcta. Nuestra solución (e) se acerca más correcta (b), que Shen [STL08] (c) o Tappen [TFA05] (d) . Los resultados de Adobe (f) se mostrarán en la presentación.*

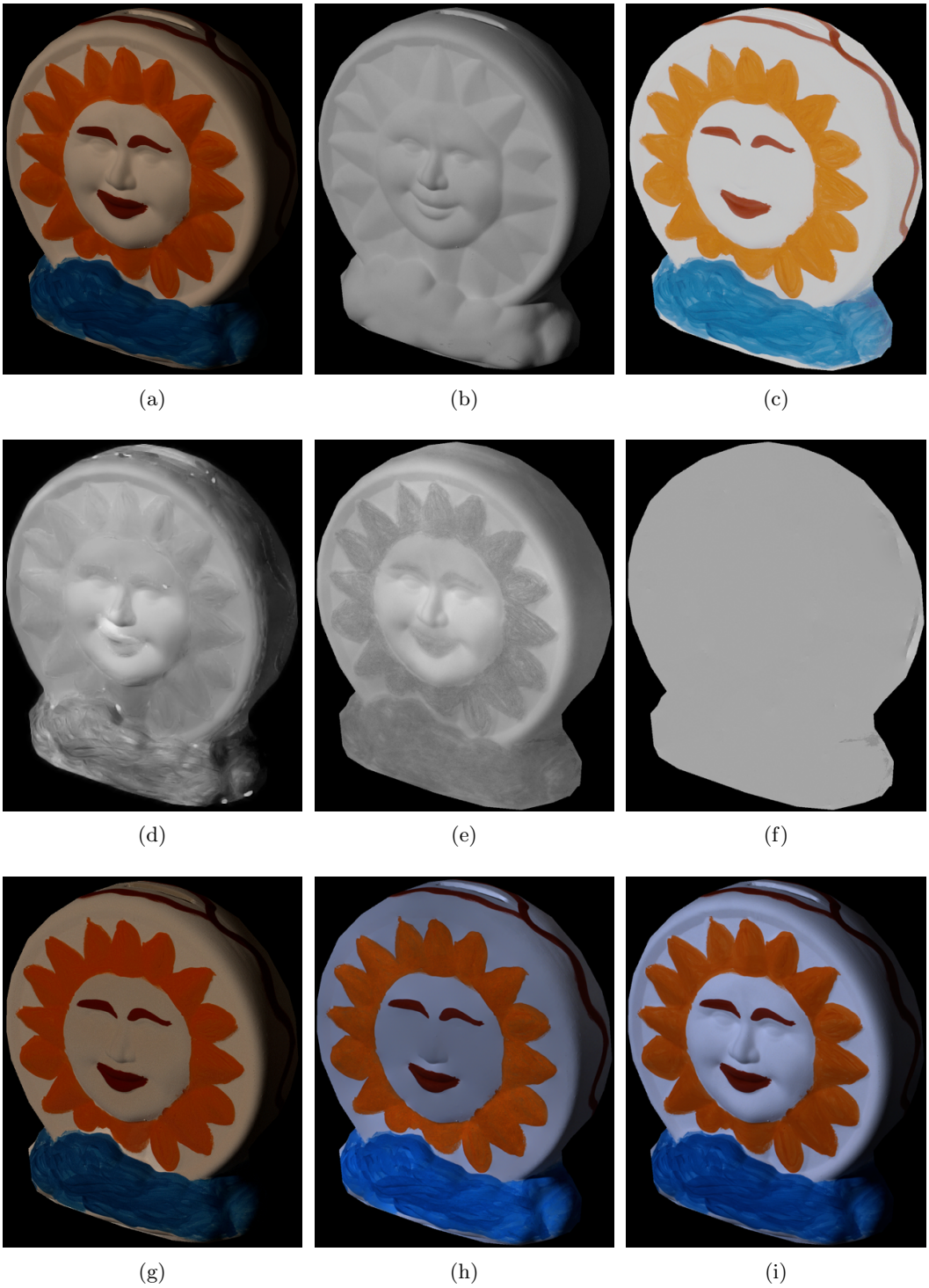
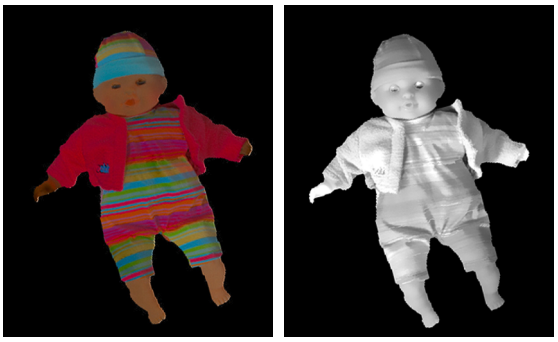


Figura 4.5: ***Comparación con otros métodos de descomposición.*** *(a) Imagen Original. (b) Iluminación real (c) Reflectancia real. (d) y (g) Iluminación y reflectancia con nuestro método (e) y (h) Iluminación y reflectancia de Shen et al. [STL08]. (f) e (i) Iluminación y reflectancia de Tappen et al. [TFA05].*

(a) Imagen original y luminancia perceptual

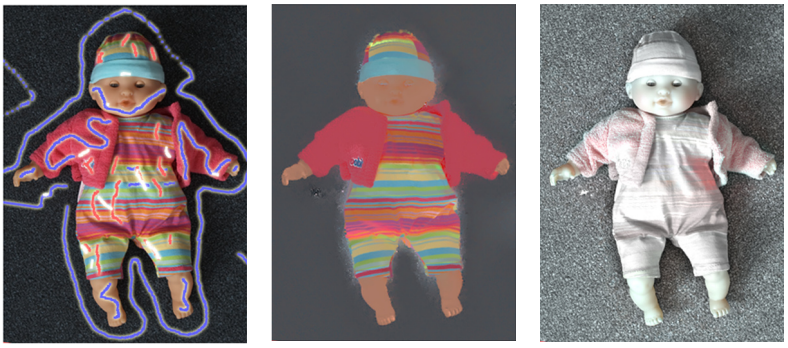


(b) Nuestra descomposición:
reflectancia e iluminación

(c) Reflectancia e iluminación de Shen [STL08]

(d) Reflectancia e iluminación de Tappen [TFA05]

(e) Reflectancia e iluminación de Weiss
a partir de 40 imágenes [Wei01]



(f) Trazos del usuario, reflectancia e iluminación de Bousseau [BDP09]

Figura 4.6: ***Comparación con otros métodos de descomposición***

Capítulo 5

Conclusiones

En este capítulo se realiza un resumen de las aportaciones de este trabajo al campo de la informática gráfica y se comenta el camino futuro de esta investigación. También recoge el resumen temporal del proyecto y las conclusiones personales de la autora.

5.1. Trabajo Realizado

Una vez finalizado el proyecto, podemos concluir que se han alcanzado los objetivos establecidos al comienzo del proyecto (ver sección 1.2):

- Se ha desarrollado un algoritmo nuevo de descomposición en imágenes intrínsecas que supera en muchos aspectos a los más relevantes en este área de investigación, tanto en algoritmos automáticos: el método de Tappen et al. [TFA05] y el método de Shen [STL08], como en algoritmos que requieren de más información: el método de Weiss [Wei01] y el método de Bousseau [BPD09]. No requiere interacción del usuario y se ejecuta en tiempo interactivo.

- Se ha implementado y adaptado un algoritmo de segmentación existente [FH04b], obteniendo una segmentación basada en regiones de albedo constante.

- Se ha comprobado que la descomposición obtenida sirve de base para otras aplicaciones de edición de imágenes: obtención de 3D o cambios en la iluminación, mejorando los resultados obtenidos hasta el momento con otras técnicas.

- El trabajo realizado ha dado lugar a un posible acuerdo de estancia en Adobe Systems Inc. en San Jose (California).

5.2. Resumen temporal del proyecto

En la Figura 5.1 se muestra el resumen de la evolución temporal del proyecto, desde su comienzo a principios de abril hasta su finalización en noviembre. A continuación se describen las fases principales:

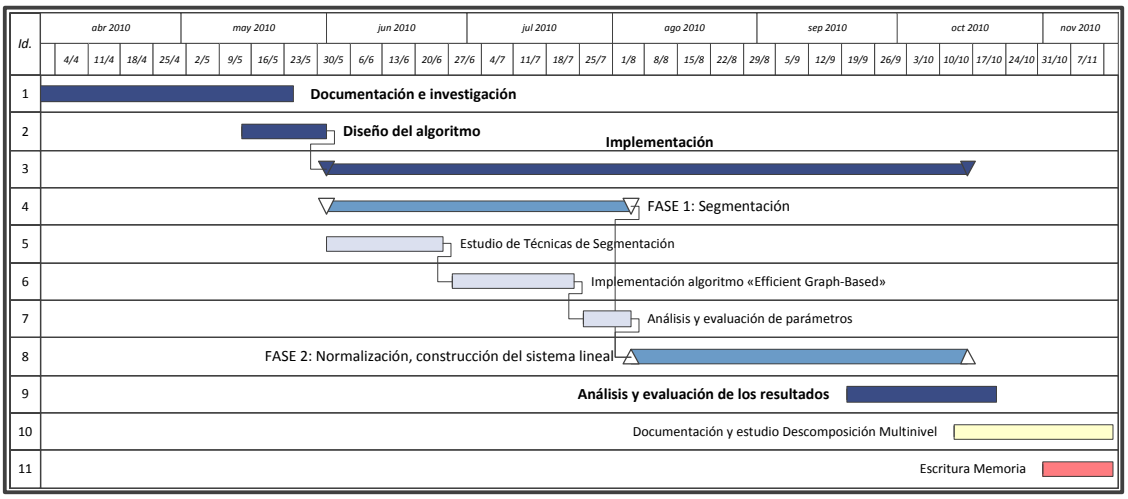


Figura 5.1: *Diagrama de Gantt con la evolución temporal del proyecto*

- **Documentación e investigación.** Una vez definido el proyecto, tuvo lugar una período de intensa documentación e investigación sobre las diversas técnicas y métodos. Aunque la tarea de investigación y lectura de documentación ha continuado durante todo el proyecto, los primeros meses fueron clave para del diseño del algoritmo.

- **Diseño del algoritmo.** Este período coincide con los últimos días de la fase inicial de investigación debido a que en ese momento ya se disponían de los conocimientos y las ideas necesarias para diseñar un buen algoritmo de descomposición.

- **Implementación.** La fase de implementación está dividida en dos fases principales:

  1. FASE 1: Segmentación. Se implementó y rediseñó el algoritmo de segmentación *Efficient Graph-Based Image Segmentation [FH04b]*. Previamente, se habían estudiado y evaluado los distintos algoritmos de segmentación existentes. Asimismo, una vez implementado el algoritmo, se estudió su comportamiento para comprobar que cumplía nuestros requisitos y se exploró el espacio de parámetros.
  2. FASE 2: Normalización y construcción del sistema lineal. Se implementó al principio de la fase un método de resolución simple del problema que, posteriormente, fue refinado gracias a la linealización del problema. A partir de ese momento, se fue perfeccionando la solución y mejorando el sistema de ecuaciones hasta dar con la solución óptima.

- **Análisis y evaluación de los resultados.** Esta fase coincide con el final la *fase 2* de la implementación, ya que el análisis de los resultados fue paralelo a la construcción y perfeccionamiento del sistema lineal. Finalmente, se compararon los resultados

con las técnicas más relevantes en descomposición de imágenes intrínsecas y se confirmó el éxito de esta investigación.

- **Documentación y estudio de la descomposición multinivel.** A partir de los resultados obtenidos se observó que se podrían añadir mejoras sustanciales al algoritmo si se incorporaba análisis multinivel de la imagen. Por ello comenzó un fase, que continúa actualmente, estudiando diversas de esas técnicas.

- **Escritura de la memoria.** Escritura del presente documento que recoge la memoria del PFC.

5.3. Trabajo Futuro

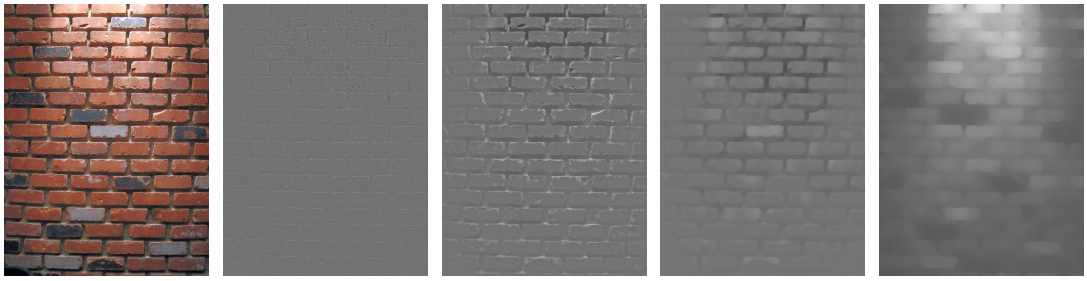
Actualmente se está trabajando en un artículo que será sometido al congreso internacional de informática gráfica SIGGRAPH 2011 y que contendrá los resultados de esta investigación.

Se está estudiando la posibilidad de incorporar técnicas de detección de luces basándonos en los estudios de Lopez-Moreno et. al [LMHRG10]. El conocimiento previo de la luces de la escena puede ser un punto clave en la segmentación de la imagen en parches de albedo constante. Esta información nos permitiría desambiguar zonas de oscuridad debidas a sombras que por ser demasiado oscuras, sean consideradas como clusters individuales y por ello generar un error en la normalización. Por otro lado, conocer la dirección de la luz nos permite estudiar la orientación de los gradientes y mejorar nuestra aproximación en base a ese conocimiento.

Otra de las principales ideas que se están explorando es la posibilidad de utilizar descomposición multi-escala [SSD09, FAR07, FFLS08], o trabajar sobre distintas resoluciones de la imagen. En concreto, se está evaluando el comportamiento de la técnica de Subr et al. [SSD09] que obtiene la descomposición de una imagen en varios niveles de detalle (ver Figura 5.2). El uso de estas técnicas nos permitiría resolver los problemas que pueden causar texturas de alto rango, es decir mucho detalle, y que por ser tratadas del mismo modo que el resto de la imagen, y ser proporcionalmente mucho más pequeñas, no desaparezcan.

5.4. Conclusiones personales

Tanto el trabajo realizado como los resultados obtenidos han sido altamente gratificantes. No sólo hemos realizado una importante aportación al campo de la investigación en informática gráfica, sino que he aprendido los últimos avances en el área y podido aplicar y consolidar los conocimientos adquiridos en la carrera. Por otra parte, la gran carga de



(a) (b) (c) (d) (e)

Figura 5.2: ***Descomposición multinivel de [SSD09]***. *En la imagen (a) vemos la imagen original. En (b)-(e) aparecen los niveles de detalle desde el más fino al más grueso. En (e) se aprecian indicios de la iluminación de la escena.*

investigación que llevaba este proyecto me ha permitido experimentar tanto la satisfacción de obtener buenos resultados, como la desesperación, en algunos casos, de estar semanas con algo que finalmente no da los resultados esperados. Después de todo, he podido comprobar que todo trabajo tiene su recompensa y en este caso, voy a tener la posibilidad de realizar una estancia en Adobe Systems Inc. en San José (California) y de publicar los resultados en uno de los congresos más importantes de este área. Personalmente, trabajar con Jorge y Diego estos meses ha sido un increíble placer tanto por el entusiasmo e interés que transmiten en su trabajo, como por lo mucho que he podido aprender de ellos.

Bibliografía

[BPD09] Adrien Bousseau, Sylvain Paris, and Frédo Durand. User assisted intrinsic images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)*, 28(5), 2009.

[BR94] Berry M. Chan T. F. Demmel J. Donato J. Dongarra J. Pozo R. Eijkhout V. Van der Vorst H. Barret, R. and C. Romine. *Templates for the Solution of Linear Systems: Building Blocks for iterative Methods, 2nd Edition*. SIAM, 1994.

[BT78] H.G. Barrow and J.M. Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, pages 3–26, 1978.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:2001, 2001.

[CM02] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603 –619, May 2002.

[CPD07] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 103, New York, NY, USA, 2007. ACM.

[Cro80] George Robert Cross. *Markov random field texture models*. PhD thesis, East Lansing, MI, USA, 1980. AAI8112063.

[EHLM71] John Edwin H. Land and J. Mccann. Lightness and retinex theory. *Journal of the Optical Society of America*, pages 1–11, 1971.

[FAR07] Raanan Fattal, Maneesh Agrawala, and Szymon Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 51, New York, NY, USA, 2007. ACM.

[FDB91] Brian V. Funt, Mark S. Drew, and Michael Brockington. Recovering shading from color images. In *ECCV-92: Second European Conference on Computer Vision*, pages 124–132. Springer-Verlag, 1991.

[FFLS08] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), August 2008.

[FH04a] Hui Fang and John C. Hart. Textureshop: texture synthesis as a photograph editing tool. *ACM Trans. Graph.*, 23:354–359, August 2004.

[FH04b] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:2004, 2004.

[FH04c] P.F. Felzenszwalb and D.R. Huttenlocher. Efficient belief propagation for early vision. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2004.

[FHLD06] Graham D. Finlayson, Steven D. Hordley, Cheng Lu, and Mark S. Drew. On the removal of shadows from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:59–68, 2006.

[GDFL04] Mark S. Drew Graham D. Finlayson and Cheng Lu. Intrinsic images by entropy minimization. In *Proc. 8th European Conf. on Computer Vision, Praque*, pages 582–595, 2004.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721 –741, 1984.

[Hor86] B. K. Horn. *Robot Vision*. MIT Press, 1986.

[I.T90] I.T.U. Basic parameter values for the hdtv standard for the studio and for international programme exchange, 1990.

[KRFB06] Erum Arif Khan, Erik Reinhard, Roland W. Fleming, and Heinrich H. Bülthoff. Image-based material editing. *ACM Trans. Graph.*, 25:654–663, July 2006.

[LB00] M.S. Langer and H.H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29:649–660, 2000.

[LMHRG10] Jorge Lopez-Moreno, Sunil Hadap, Erik Reinhard, and Diego Gutierrez. Compositing images through light source detection. *Computers and Graphics*, In Press, Corrected Proof, 2010.

[LMJH+] Jorge Lopez-Moreno, Jorge Jimenez, Sunil Hadap, Ken Anjyo, Erik Reinhard, and Diego Gutierrez. Non-photorealistic, depth-based image editing. *Computers and Graphics*.

[LWQ+08] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, pages 1–9, 2008.

[MK10] Branislav Micusik and Jana Kosecká. Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision*, 89:106–119, 2010. 10.1007/s11263-010-0327-9.

[MTC07] Ankit Mohan, Jack Tumblin, and Prasun Choudhury. Editing soft shadows in a digital photograph. *IEEE Comput. Graph. Appl.*, 27(2):23–31, 2007.

[NS94] N.M. Nachtigal and B.D. Semeraro. Qmr methods in computational fluid dynamics. 1994.

[PJ89] T.N. Pappas and N.S. Jayant. An adaptive clustering algorithm for image segmentation. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 1667 –1670 vol.3, May 1989.

[RT10] R. Raskar and J. Tumblin. *Computational Photography: Mastering New Techniques for Lenses, Lighting, and Sensors*. A K Peter, 2010.

[SM00] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888 –905, August 2000.

[SSD09] Kartic Subr, Cyril Soler, and Frédo Durand. Edge-preserving multiscale image decomposition based on local extrema. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)*, Annual Conference Series. ACM, ACM Press, dec 2009.

[STL08] Li Shen, Ping Tan, and Stephen Lin. Intrinsic image decomposition with non-local texture cues. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–7, 2008.

[SZS+08] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):1068 –1080, 2008.

[TFA05] Marshall F. Tappen, William T. Freeman, and Edward H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1459–1472, 2005.

[UPH07] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):929 –944, 2007.

[Wei01] Yair Weiss. Deriving intrinsic images from image sequences. *Computer Vision, IEEE International Conference on*, 2:68, 2001.

[WH97] Guo-Qing Wei and Gerd Hirzinger. Parametric shape-from-shading by radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:353–365, 1997.

[WTBS07] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Natural shadow matting. *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 26(2):8, 2007.

[YFW03] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. pages 239–269, 2003.

[Zah71] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20:68–86, January 1971.

Apéndices

Apéndice A

Estudio de la segmentación

Segmentar una imagen consiste en dividirla en las distintas partes que la integran. A simple vista parece un problema sencillo, pero decidir qué es una buena o mala segmentación depende mucho de la finalidad de la aplicación y de lo que consideremos perceptualmente importante. Podemos querer segmentar la imagen en grandes regiones que identifiquen objetos para su reconocimiento, o buscar conjuntos de pixels de color o textura homogéneos que puedan servir para otras técnicas de visión. Sin embargo, hay un conjunto de características que siempre son deseables para una buena segmentación: las regiones obtenidas deben ser homogéneas y uniformes respecto a alguna propiedad, como puede ser el tono o el color, y, las regiones adyacentes deben ser considerablemente diferentes respecto a la propiedad en la que son uniformes.

A.1. Técnicas de segmentación

Nuestra idea de qué es una buena segmentación se basa en las nociones anteriores. En concreto, queremos localizar conjuntos de pixels (clusters) de reflectancia o albedo constantes. Por ello, siguiendo la aproximación de Funt et al. [FDB91], vamos a buscar variaciones de reflectancia en base a las variaciones de crominancia, basándonos en que las variaciones de sombreado no alteran la cromaticidad. Siguiendo esta premisa y con la necesidad de un algoritmo eficiente, ya que nuestra aplicación está orientada al procesamiento de imágenes en tiempo interactivo, evaluamos varias técnicas de segmentación existentes. En concreto, evaluamos algoritmos basados en Campos Aleatorios de Markov, métodos de agrupamiento (o *clustering*) y métodos basados en grafos.

A.1.1. Algoritmos basados en campos Aleatorios de Markov

Los campos aleatorios son un tipo de modelo estadístico que proporcionan medidas de probabilidad sobre dominios de definición que tienen relaciones de tipo espacial o temporal [Cro80], como por ejemplo una imagen. Se basan en probabilidades bayesianas y pueden resolver diversos problemas de etiquetado: segmentación, visión estereo, elimi-

nación de ruido, reconstrucción de texturas, etc. Según formuló Geman et al. [GG84], los campos aleatorios de Markov proporcionan un buen modelo teórico para algunos de estos problemas de inferencia en imágenes, en los que queremos obtener *lo que hay realmente*, a partir de los datos que disponemos que, en estos casos, son matrices que representan los píxeles de la imagen.

Asumimos que disponemos de un conjunto de observaciones sobre la imagen y_i, y que queremos inferir su valor latente en la escena x_i (el índice i puede representar un pixel o una región de píxeles). Además, asumimos que hay una dependencia estadística entre x_i e y_i que viene definida por una función de compatibilidad $\phi(x_i, y_i)$. Por otra parte, las variables de la imagen también están relacionadas de tal manera que podemos establecer otra función de compatibilidad $\psi(x_i, x_j)$ que relaciona pares de pixels vecinos (ver Figura A.1). Dicho esto, podemos formular probabilidad condicional entre la imagen observada y_i y la imagen que queremos inferir con lo siguiente:

$$p(x, y) = \frac{1}{Z} \prod \psi(x_i, x_j) \prod \phi(x_i, y_i) \tag{A.1}$$

donde Z es una constante de normalización.

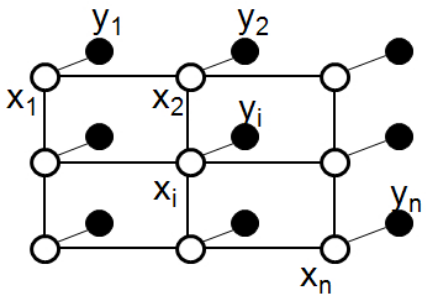


Figura A.1: *Ejemplo gráfico de un campo aleatorio de Markov. Los puntos negros y_i representan las observaciones y los puntos blancos x_i los valores que buscamos.*

En otras palabras, los campos aleatorio de Markov establecen que la probabilidad condicional de que un pixel tenga un determinado valor viene dada por el valor de sus vecinos, no por la imagen entera, y por tanto, pueden ser usados para modelar determinadas propiedades de continuidad y suavidad entre regiones de la imagen. Existen numerosas técnicas que resuelven directamente problemas planteados en base a campos aleatorios de Markov. Algunas de las más usadas y eficientes [SZS+08] son la propagación del conocimiento (*Belief Propagation* [YFW03, FH04c] o el corte de grafos (*Graph Cuts*) [BVZ01]. Sin embargo, estos métodos, aunque eficientes, necesitan conocer el número de regiones de la imagen o al menos disponer de alguna función de energía que relacione las variables y las observaciones para poder definir las funciones de compatibilidad. En nuestro caso, no hemos experimentado con estas técnicas puesto que hemos tenido conocimiento de que Adobe ya ha desarrollado una línea de investigación explorando ese camino y no obtienen buenos resultados sin intervención del usuario.

A.1.2. Métodos de clusterizado

Una de las técnicas más usadas en algoritmos de visión por computador es la llamada desplazamiento de media (*mean shift*) [CM02]. Se encuentra dentro de las técnicas que tratan de buscar clusters dentro de un espacio de características (no tienen en cuenta relaciones espaciales entre los pixels) y las cuales asumen que la imagen es constante por segmentos. El algoritmo de Mean-shift realiza un suavizado de la imagen agrupando píxeles semejantes y formando clusters que se identifican por su color más significativo. Posteriormente, un refinamiento de este primer clusterizado obtiene la segmentación deseada de la imagen. Esta técnica obtiene buenos resultados, pero como se dice en [UPH07], es muy sensible a los parámetros, es decir, la elección de los parámetros de ejecución del algoritmo depende mucho de la imagen y de los resultados que queramos obtener, motivo por el cual no sirve a nuestros propósitos.

Hay que hacer mención especial a una técnica de clusterizado denominada *k-means* adaptativo [PJ89]. Ésta combina la idea sencilla de trabajar en un espacio de características donde los pixels están relacionados por color, al igual que *mean shift*, junto con diversas propiedades de continuidad espacial. La idea se asemeja, en cierto modo, a la que usamos en nuestro método por lo que no descartamos en trabajo futuro evaluar su comportamiento en nuestro algoritmo.

A.1.3. Métodos basados en grafos

Las técnicas basadas en grafo, normalmente representan los píxeles de la imagen como un grafo ponderado no dirigido, donde cada punto representa un nodo y el peso viene determinado por alguna relación entre los vértices que conecta, como puede ser la diferencia de intensidades. Un conjunto de técnicas muy extendidas son las basadas en realizar cortes mínimos en grafos, donde el criterio de corte esta diseñado para minimizar la similitud entre los píxeles que están siendo divididos. La técnica más famosa es la conocida como cortes normalizados (*normalized cuts*) [SM00] y destaca porque en lugar de solo capturar propiedades locales de la imagen, encuentra características a nivel global. No obstante, estas aproximaciones de cortes normalizados son demasiado lentas para nuestro algoritmo.

Dentro de estas técnicas basadas en grafo se encuentra la que hemos escogido en nuestro algoritmo y que está siendo usada recientemente en aplicaciones de visión estéreo para la búsqueda de *superpixels* [MK10]. En la siguiente sección se da una descripción detallada de la misma.

A.2. Segmentación eficiente basada en grafo

El método elegido, ideado por Felzenszwalb y Huttenlocher [FH04b], se trata de un algoritmo basado en grafo que se ajusta muy bien a nuestras necesidades de rapidez y

eficacia. La clave de su utilidad es que dispone de un umbral adaptativo como veremos a continuación. El algoritmo parte un grafo no dirigido $G = (V, E)$ formado por un conjunto n de vértices $v \in V$, que se corresponden con los pixels a segmentar de la imagen representados en el espacio de características, y un conjunto m de aristas $\{e_i\}$ que constituyen pares de vértices vecinos. Cada arista tiene tiene un peso $w((v_i, v_j))$, que representa la similitud entre los dos pixels conectados por esa arista. La segmentación final será $S = (C_1, ..., C_r)$ donde C_i es un cluster de puntos. El algoritmo es el siguiente:

1. Ordenar el conjunto de aristas $E = (e_1, ..., e_m)$ tales que $|e_t| \leq |e_{t'}| \forall t < t'$.
2. Sea $S^0 = (\{v_1\}, ..., \{v_n\})$, (inicialmente cada cluster contiene exactamente un vértice).
3. For $t = 1, ..., m$
   a Sean v_i y v_j los vértices conectados por e_t.
   b Sea $C_{v_i}^{t-1}$ el componente que contiene el punto v_i en la iteración $t-1$ y $l_i = max_{mst} C_{v_i}^{t-1}$ el mayor peso de las aristas que hay dentro de $C_{v_i}^{t-1}$. Del mismo modo obtenemos l_j.
   c Uniremos los componentes $C_{v_i}^{t-1}$ y $C_{v_j}^{t-1}$ si,

$$|e_t| < \text{mín} \left\{ l_i + \frac{k}{|C_{v_i}^{t-1}|}, l_j + \frac{k}{|C_{v_j}^{t-1}|} \right\} \tag{A.2}$$

   donde k es una constante.
4. $S = S^m$

En lugar de usar un valor fijo para determinar el umbral a partir del cual dos regiones se consideran distintas, del mismo modo que lo hace Zahn en su método [Zah71], utiliza un valor variable en la Fórmula A.2. Este umbral permite que dos componentes se unan si la arista de menor peso que los une, es menor que la máxima arista en cada uno de los componentes más el término $\tau = k/|C_{v_i}^{t-1}|$. Como vemos, τ depende del tamaño del componente y de una constante inicial k. En la primera iteración del algoritmo $l_i = l_j = 0$, y $|C_{v_i}^0| = |C_{v_j}^0| = 1$, por tanto, k inicialmente representa el máximo peso de arista que podrá ser añadido a cada componente, $k = l_{max}$. Al aumentar el número de puntos por componente, la tolerancia de añadir nuevas aristas disminuye y se realizan menos uniones. Intuitivamente, k controla el tamaño final de los clusters ya que controla las primeras iteraciones de la segmentación.

Tipos de grafo: KNN vs Grid

En el artículo original se proponen dos estructuras de grafo distintas, una basada en una malla (grafo *GRID*), donde cada pixel está conectado con sus 8-vecinos más próximos por posición y otra basada en el método de vecino más próximo (grafo *KNN*) donde se realiza un mapeo de cada pixel en un espacio nuevo de características y donde se puede

definir libremente el número de vecinos a usar.

En el caso de usar un grafo *GRID*, la función que define la similitud entre los dos pixels conectados por una arista, viene dada por su diferencia de color. Como propone el autor, usamos la distancia Euclídea L_2,

$$w((v_i, v_j)) = \|C(v_i) - C(v_j)\| = \sqrt{\sum_{t=1}^{N} |C(v_i)_t - C(v_j)_t|} \tag{A.3}$$

donde $C(v)$ es el vector de color del vértice v, siendo $C(v) = \{r, g, b\}$ en espacio de color *RGB* y $C(v) = \{a, b\}$ en espacio de color *Lab*.

En el otro caso, usando grafo *KNN* se realiza un mapeo de cada vértice en el espacio $\{x, y, C(x, y)\}$, donde (x, y) es la localización del vértice en la imagen y $C(x, y)$ representa el color del punto que dependerá del modelo usado. Del mismo modo que con el grafo *GRID* usamos la distancia Euclídea L_2 para definir los pesos de las aristas, pero ahora también tiene influencia la posición del pixel en la imagen. La ventaja de usar KNN frente a GRID, es que en el primero, el poder seleccionar un número variable de vecinos y capturar en la función de similitud la posición junto con el color, permite que las conexiones de los píxeles sean más flexibles pudiendo conectarse regiones físicamente separadas. En cambio, usando grafo GRID tenemos una estructura rígida en la que solo podemos realizar conexiones locales.

La influencia del modelo de color: RGB vs Lab

El artículo original realiza la segmentación de la imagen usando el espacio de color RGB. A pesar de que los resultados son buenos, no sirven totalmente para nuestros propósitos. Al trabajar sobre el espacio de color RGB, tratan del mismo modo todos los pixels de la imagen y no tienen en cuenta que en una región pueda haber variaciones de luminosidad producidas por sombreado. Por ello, seguimos los estudio de Funt et al. [FDB91] y decidimos utilizar el modelo de color Lab que, por la forma en que está definido, nos permite abstraernos de las variaciones de luminosidad en el color de los materiales y trabajar directamente con la cromaticidad.

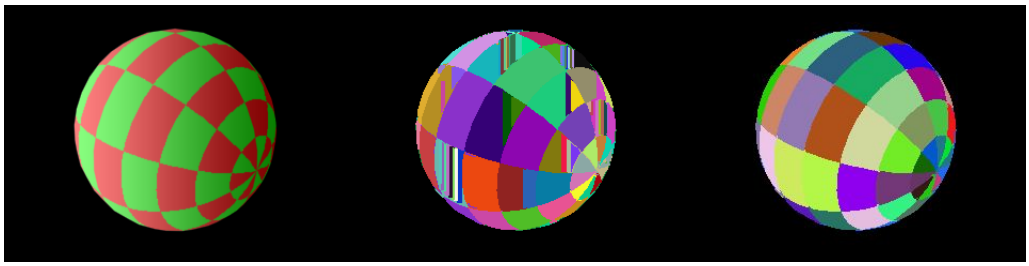
El modelo de color *Lab* caracteriza cada color con la ayuda de un parámetro de intensidad correspondiente a la luminancia y de dos parámetros de crominancia que describen el color. Ha sido especialmente estudiado para que las distancias calculadas entre colores correspondan a las diferencias percibidas por el ojo humano. En concreto, los tres parámetros están definidos del siguiente modo:

1. La componente L es la luminosidad, que va de 0 (negro) a 100 (blanco).

2. La componente a representa la gama de rojo (valor positivo) a verde (negativo) pasando por el blanco (0) si la luminosidad vale 100.

3. La componiendo b representa la gama de amarillo (valor positivo) a azul (negativo) pasando por el blanco (0) si la luminosidad vale 100.

Por su parte, RGB es un modelo de color basado en la síntesis aditiva, con lo que cada color viene representado mediante la mezcla por adición de los tres colores primarios: rojo, verde y azul. La diferencia entre usar uno u otro modelo, en nuestro caso, es sustancialmente importante ya que con Lab conseguimos que una región con un albedo constante, pero sometida a un foco de luz y por ello con un gradiente de intensidad, sea segmentada como un único cluster y no como varios como ocurre erróneamente en los ejemplos de las Figuras A.2 y A.3 para el caso de RGB.



(a) Imagen original (b) Segmentación RGB (c) Segmentación Lab

Figura A.2: ***Comparación de segmentación RGB vs Lab.*** *Los mejores resultados se obtienen con espacio de color Lab*

A.2.1. Experimentos realizados

En los experimentos hemos realizado pruebas modificando el valor del umbral adaptativo k y probando distintos tipos de grafos. En las Figuras A.5 y A.6 se pueden ver los resultado para dos imágenes muy distintas y que en sí, abarcan la mayoría de los casos que vamos a tratar. La primera se trata de una imagen donde el tamaño de los clusters es grande en relación con el tamaño de la imagen y hay poca variación cromática apreciable a simple vista. Por el contrario, la segunda imagen tiene un tamaño de clusters en proporción mucho menor y las diferencias de cromaticidades son mucho más pronunciadas.

Si observamos los resultados de estas podemos comprobar como el nivel de detalle de la segmentación aumenta según disminuye el valor del umbral. Esto se debe a que este valor controla, en las primeras iteraciones de la segmentación, el peso máximo que puede tener una arista para formar parte de un cluster (ver Fórmula A.2). Es decir, mide la máxima diferencia cromática que pueden tener dos pixels que se encuentren en el mismo cluster. Según aumenta el tamaño de los clusters, este umbral deja de tener tanto peso y el criterio para decidir si un pixel pertenece o no a un cluster viene dado por la propia coherencia

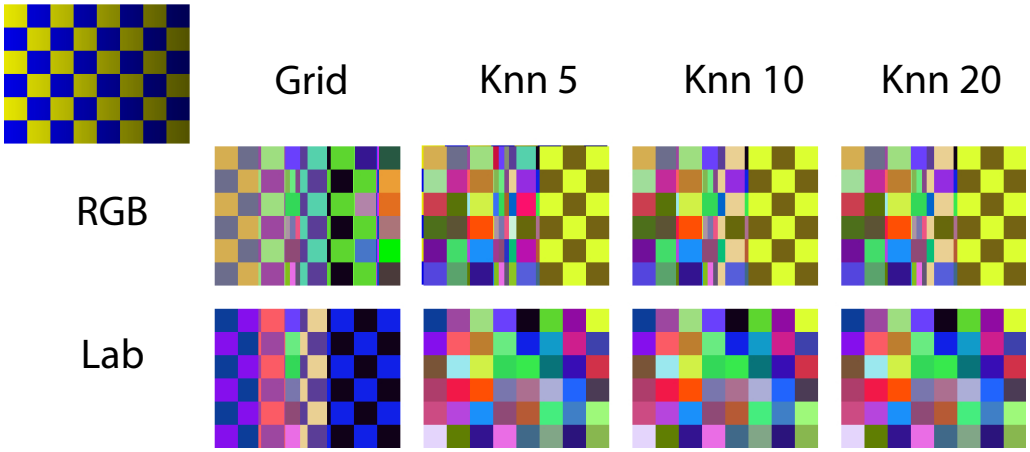
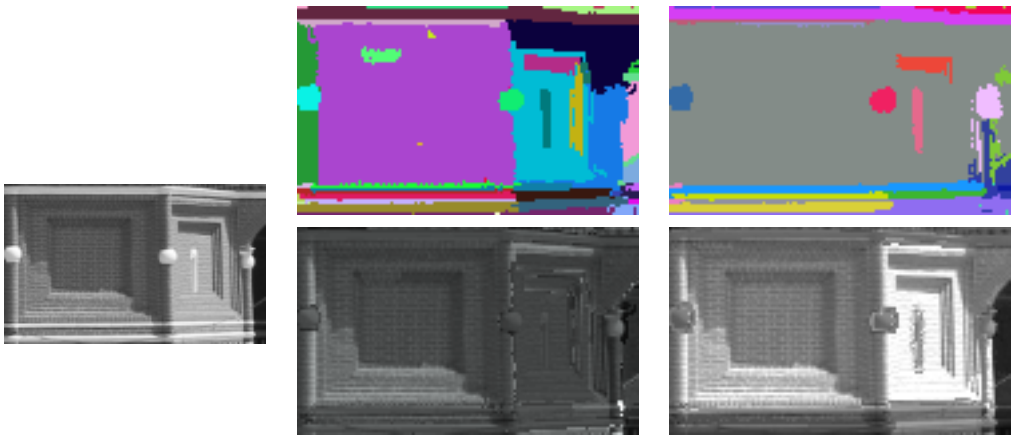


Figura A.3: ***Comparación de segmentación RGB vs Lab.*** *Los mejores resultados se obtienen con grafo KNN y espacio de color Lab*

interna del mismo. Para nuestro propósito, puesto que necesitamos que los clusters identifiquen correctamente fragmentos de color constante y no demasiado pequeños, un umbral de 25 o 50 resultaría válido (ver Figuras A.5 y A.6). Respecto al tipo de grafo a usar, comprobamos que los resultados son muy parecidos en todas las versiones del grafo KNN. No obstante, sí se aprecia diferencia respecto a usar grafo GRID, ya que este último ha cometido errores no deseables en la segmentación: ha divido regiones de la imagen de albedo constante pero con variaciones de iluminación que deberían haberse considerado como una sola. Hemos considerado aceptable la segmentación con grafo KNN y cinco vecinos y hemos realizado una segunda fase de pruebas.

Partiendo de los primeros resultados obtenidos, hemos realizado otro conjunto de pruebas para asegurarnos de elegir los parámetros óptimos. En la Figura A.7 vemos los resultados de la segmentación fijando el tipo de grafo a KNN con 5 vecinos y modificamos el umbral con valores de 25, 50 y 75. Como ya habíamos pensado, la segmentación más correcta se da para un umbral de 50, obteniendo una segmentación muy poco detallada según aumentamos de valor. Asimismo, en la Figura A.8 vemos los resultados fijando el valor del umbral a 50 y variando el tipo de grafo. A simple vista, puede parecer que la segmentación con grafo GRID es mejor ya que obtenemos un mayor número de clusters, sin embargo, si ejecutamos el algoritmo de normalización sobre estas imágenes podemos comprobar que no es así. En la Figura A.4 vemos un ejemplo de error de segmentación con grafo GRID y por tanto error en la normalización final. Este problema es debido a que usando grafo GRID tenemos una estructura rígida: cada pixel se conecta con sus 8-vecinos más próximos. En cambio, el grafo KNN realiza las conexiones de tal modo que se agrupan los pixels más parecidos en color a pesar de que no estén físicamente conectados. Los resultados en KNN son muy similares para los tres casos, por tanto, decidimos usar el grafo KNN con cinco vecinos ya que el tiempo de computación es inferior a usar diez o treinta.



(a) (b) (c)

Figura A.4: ***Error en la normalización.*** *La imagen (a) representa la luminancia original. En (b) vemos, en la parte superior la segmentación con grafo GRID y en la parte inferior la normalización final. Podemos comprobar como esta es errónea ya que no mantiene el gradiente de iluminación. En (c) tenemos el resultado usando grafo KNN. En este caso sí se han mantenido los gradientes de iluminación de manera correcta.*

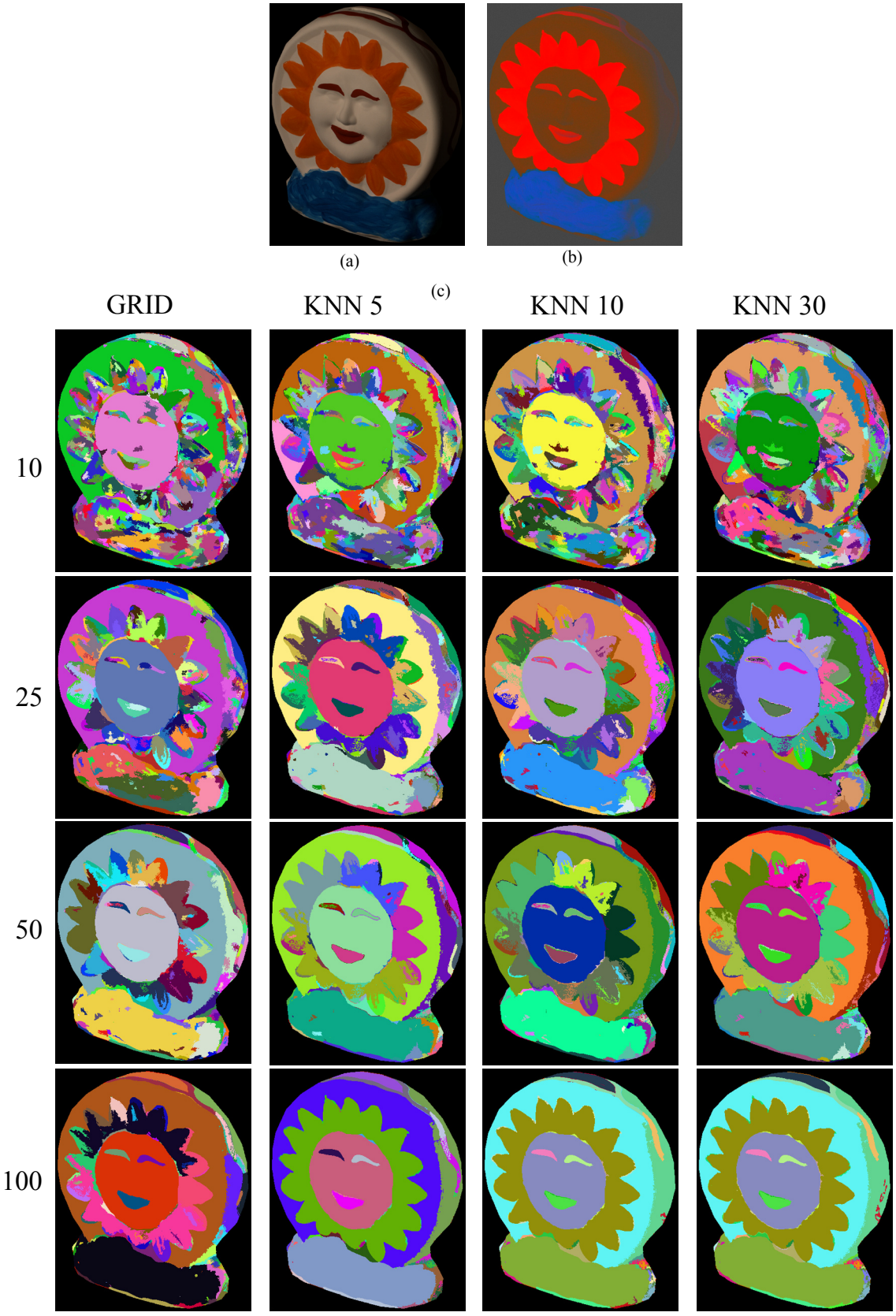


Figura A.5: ***Exploración de parámetros en la segmentación***

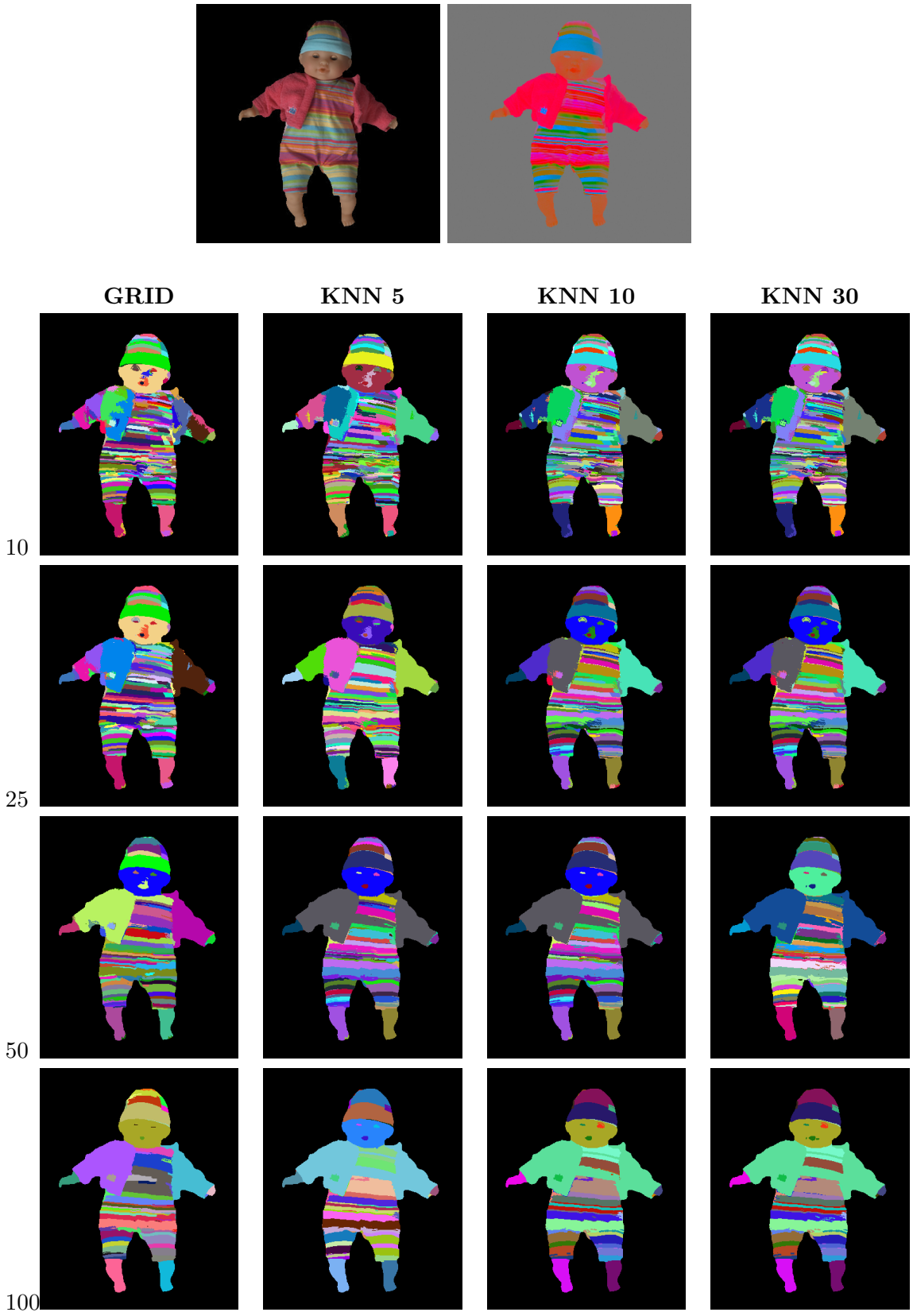


Figura A.6: ***Exploración de parámetros en la segmentación***

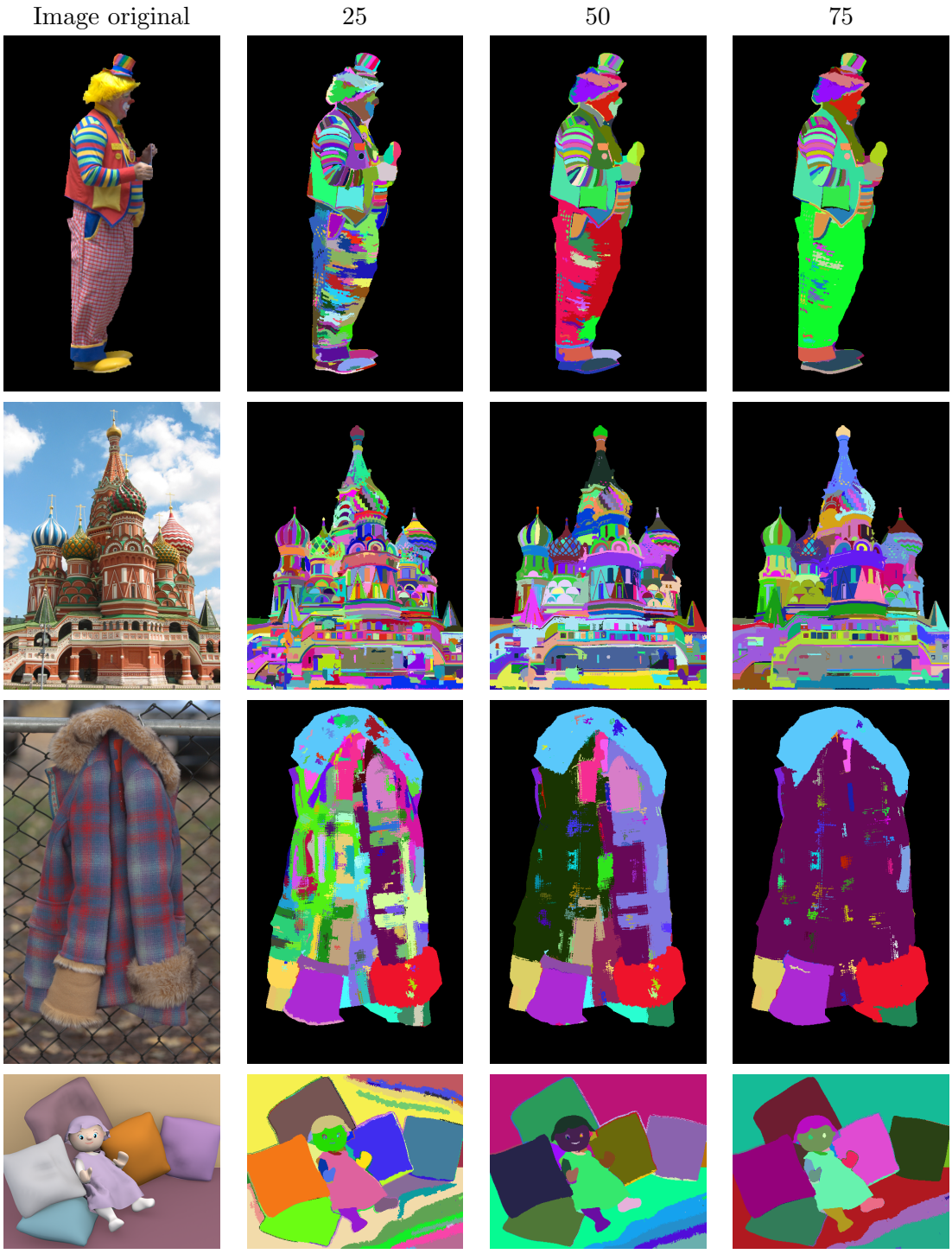


Figura A.7: ***Resultados segmentación (parte 1).*** *La primera imagen representa la imagen original y las siguientes son resultados de la segmentación utilizando un grafo KNN con 5 vecinos y modificando el umbral adaptativo con los valores: 10, 25 y 50.*

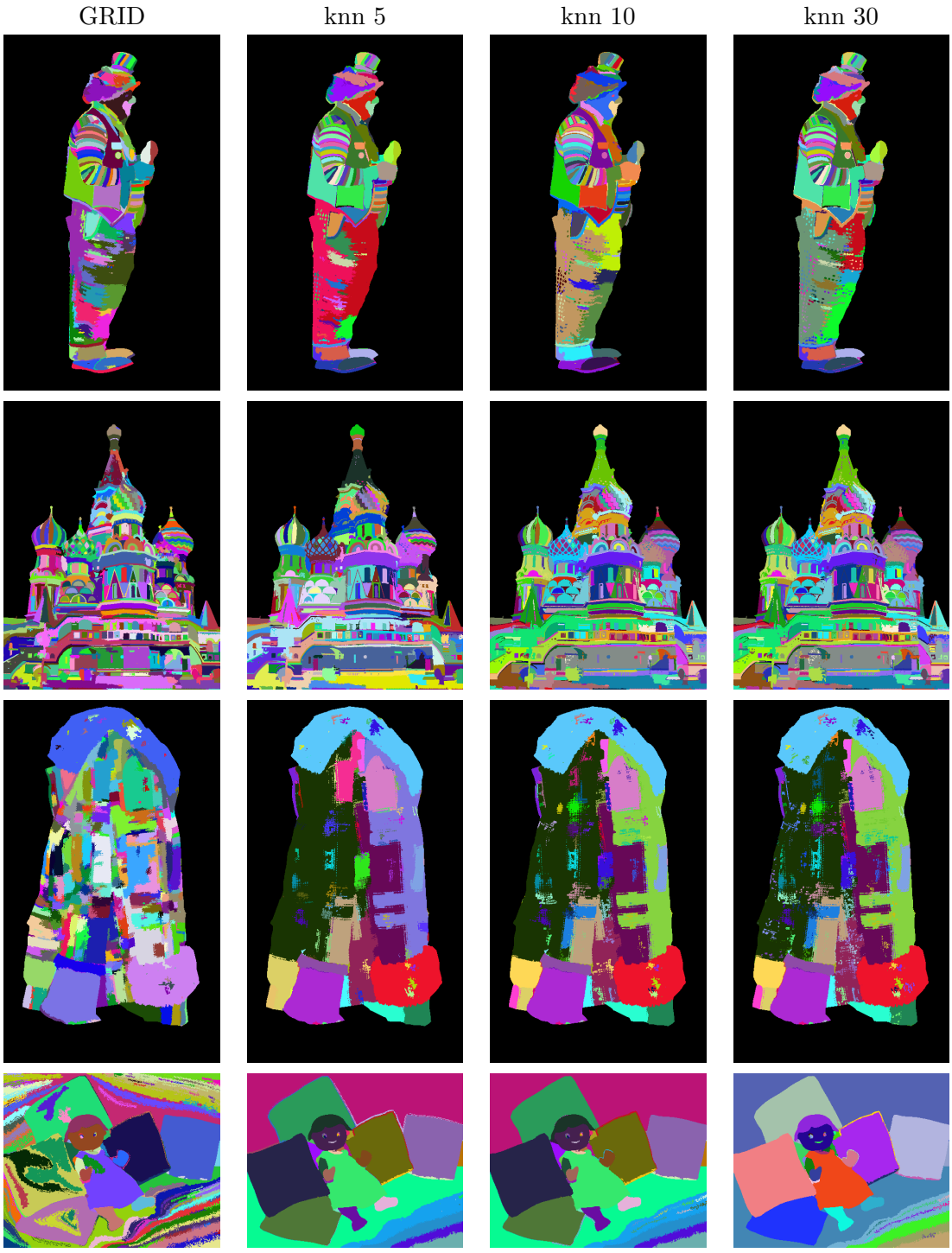


Figura A.8: ***Resultados segmentación (parte 2).*** *En estos resultados el umbral adaptativo tiene un valor fijo igual a 50. De izquierda a derecha variamos el tipo de grafo usado: grafo GRID, grafo KNN con 5 vecinos, grafo KNN con 10 vecinos y grafo KNN con 30 vecinos.*

Apéndice B

Estudio de precondicionadores

En este apéndice se realiza un estudio sobre cómo precondicionar el sistema de ecuaciones lineales definido en la sección 3.2 para conseguir mejores resultados en menos iteraciones.

B.1. Precondicionar para disminuir el número de iteraciones

Comúnmente, precondicionar un sistema lineal consiste en transformar el sistema original $Ax = b$ en un sistema equivalente $\tilde{A}x = \tilde{b}$ para reducir el número de iteraciones requerido para la convergencia. La resolución del sistema lineal equivalente no deberá transformar la solución que se obtendría con el original. La matriz $\tilde{A}$ y el vector $\tilde{b}$ se obtienen multiplicando, antes o después, A y b por una matriz M que denominamos precondicionador. Por medio del precondicionador conseguimos condiciones espectrales más favorables y se reduce el número de iteraciones requeridas para la convergencia, sin incrementar significativamente la cantidad de cálculos por iteración. En nuestro sistema lineal, utilizando el precondicionador de *Jacobi* M, donde $M = D = diag(A)$ reducimos el número de iteraciones que necesita nuestro sistema a menos de la mitad, tanto para el modelo de energía como el de flujos.

B.2. Precondicionar para mejorar la solución

Partiendo de la idea de precondicionador comentada en la sección anterior, nosotros hemos diseñado nuestro propio precondicionador, basándonos en el conocimiento que disponemos del problema, para obtener una mejor solución.

Precondicionando el sistema de energías

Retomamos el sistema lineal basado en energías $Ax = b$ planteado en la sección 3.2 en el cual buscamos normalizar las luminancias en las fronteras de los clusters. Disponemos de N clusters y M pares de clusters vecinos, ycada fila a_i de A viene definida por:

$$\forall i \in 1..M, a_i = \begin{cases} \exists k, l \in 1..N \ni a_{ik} = L_m(c_k)_{c_l},\ a_{il} = -L_m(c_l)_{c_k} \\ \text{con } k < l \wedge c_k \text{ es adyacente a } c_l \\ \\ a_{ih} = 0, \forall h \in 1..N \ni h \neq k \wedge h \neq l \end{cases} \tag{B.1}$$

$$i = M + 1, \forall j \in 1..N, a_{ij} = L_{Me}(c_j)$$

Y X y B definidos por,

$$X_{N\times 1} = \begin{pmatrix} F_{c_1} \\ F_{c_2} \\ \vdots \\ F_{c_N} \end{pmatrix} \qquad B_{(M+1)\times 1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \sum_{i=1}^{N} L_{Me}(c_i) \end{pmatrix} \tag{B.2}$$

Tal cual está planteado el sistema, todas las ecuaciones tienen la misma relevancia. Esto implica que en los casos en que la segmentación no sea buena y obtenga clusters demasiado pequeños (en concreto los clusters de sombra) habrá problemas en la resolución. Para solucionar esto, precondicionamos la matriz A con una matriz de pesos definida de tal modo que las ecuaciones que contengan pares de clusters más significativos (más grandes o con más conexiones) tengan preferencia en la solución y se normalicen preferiblemente antes que otros. Hay que recordar que el método iterativo QMR no garantiza una solución, hay ocasiones en que, por existir demasiadas conexiones o clusters el sistema no converge a la solución óptima. Añadiendo el precondicionador de pesos, estamos *diciendo* al sistema que si tiene que dejar alguna pareja sin normalizar para converger, deje la menos relevante en la imagen. Definimos el precondicionador de pesos W de la siguiente manera:

$$W = \begin{pmatrix} w_1 & 0 & ... & 0 \\ 0 & w_2 & ... & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & ... & 0 & w_{M+1} \end{pmatrix} \tag{B.3}$$

$$\begin{aligned} &\forall k \in 1..M \quad w_k = Area_{c_i} \frac{nconex_{ij}}{nconex_i} + Area_{c_j} \frac{nconex_{ij}}{nconex_j} \\ &k = M + 1 \quad w_k = \frac{\sum_{l=1}^{M} w_l}{M} \end{aligned} \tag{B.4}$$

donde c_i y c_j son los clusters conectados en la fila a_k de la matriz A, $nconex_{ij}$ representa el número total de aristas que unen los clusters c_i y c_j y $nconex_i$ representa el total de conexiones que tiene el cluster c_i con el resto de clusters de la imagen.

De este modo ponderamos cada ecuación (cada fila de la matriz A), por la importancia de esa pareja de clusters en relación al tamaño/número de conexiones con respecto al resto de la imagen. Como paso previo a utilizar el número de conexiones y el tamaño del cluster se realizaron pruebas teniendo cuenta cada característica por separado, sin embargo, no se consiguieron resultados satisfactorios.

La utilización de este precondicionador en el sistema de energías no garantiza obtener la solución correcta. Como se mostraba en la sección 3.2, es necesario replantear el problema desde otra perspectiva. Sin embargo, sí que se observan mejoras en determinados sistemas sencillos donde el número de clusters no es muy elevado, menos de cien, y las relaciones entre ellos no son muy complejas.

Precondicionando el sistema de flujos

Retomando el problema según el modelo de flujos, lo primero en lo que tenemos que fijarnos es en el cambio de productos a sumas. En este modelo, las luminancias son aditivas y no multiplicativas como en el caso anterior. Recordamos la ecuación principal que regula el sistema:

$$\ln(L_m(c_i)_{c_j}) - \ln(L_m(c_j)_{c_i}) = \varphi_j - \varphi_i \tag{B.5}$$

donde $\varphi_i = \ln(F_{c_i})$ y $\varphi_j = \ln(F_{c_j})$.

Y el propio sistema $Ax = b$, donde A es una matriz de N columnas y $M + 1$ filas con M el número de pares de clusters adyacentes y N el número de clusters de la imagen. En este caso, cada fila a_i de A viene definida por:

$$\forall i \in 1..M, a_i = \begin{cases} \exists k, l \in 1..N \ni a_{ik} = 1,\ a_{il} = -1 \\ \text{con } k < l \wedge c_k \text{ es adyacente a } c_l \\ \\ \forall h \in 1..N,\ a_{ih} = 0,\ \text{si } h \neq k \wedge h \neq l \end{cases} \tag{B.6}$$

$$i = M + 1, \forall j \in 1..N, \quad a_{ij} = 1$$

Y X y B definidos por,

$$X^T_{N\times 1} = \begin{pmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_N \end{pmatrix} \tag{B.7}$$

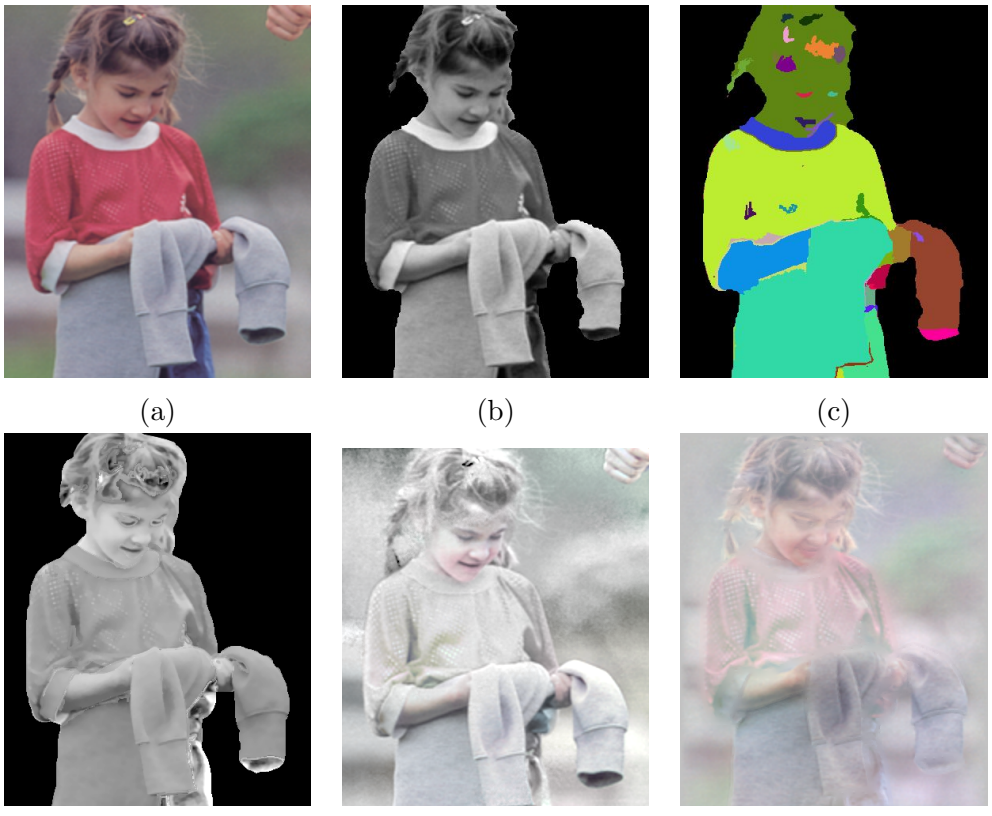
$$\begin{aligned} &B^T_{(M+1)\times 1} = \begin{pmatrix} b_1 & \dots & b_M & 0 \end{pmatrix} \text{ donde,} \\ &\forall i \in 1..M, \quad \exists k, l \in 1..N \ni b_i = \ln(L_{Me}(c_l)) - \ln(L_{Me}(c_k)) \\ &\text{con } k < l \wedge c_k \text{ es adyacente a } c_l \end{aligned} \tag{B.8}$$

Como podemos comprobar, si ponderamos cada ecuación con un peso relativo al tamaño o al número de conexiones del mismo modo que en el caso anterior, estamos mezclando factores multiplicativos con sumas de logaritmos, lo cual afecta directamente a la formulación del problema y cambia su significado inicial. Por ello, hasta el momento no hemos desarrollado un precondicionador bueno para el sistema de flujos, en su lugar, para mantener el equilibrio se añadieron las ecuaciones de equilibrio de luminancias que evitan muchos de los problemas mencionados. No obstante, se está trabajando en mejorar este sistema con algún precondicionado que no modifique la formulación inicial.

Apéndice C

Resultados y aplicaciones

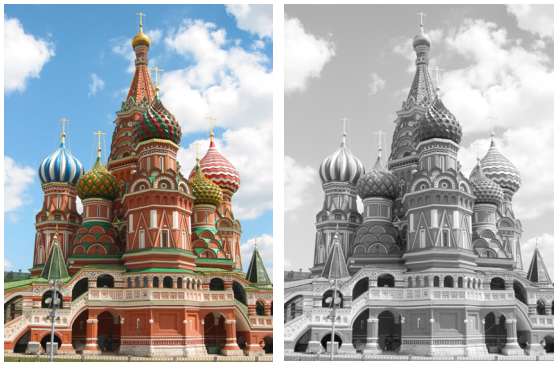
C.1. Resultados



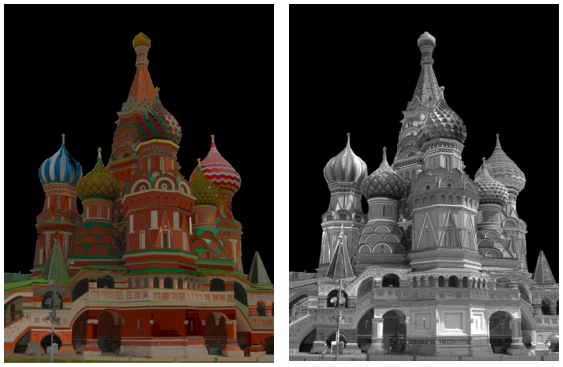
(a) (b) (c)

(d) (e) (f)

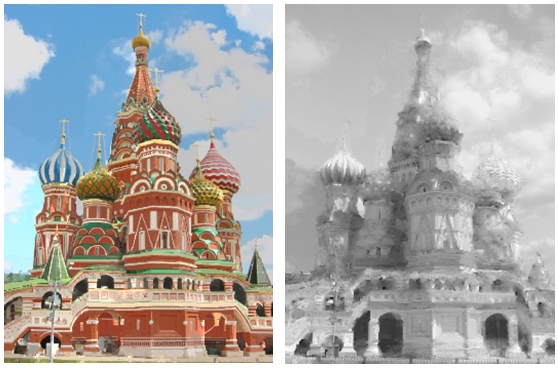
Figura C.1: ***Comparación con otros métodos de descomposición.*** *(a) Imagen Original. (b) Luminancia original. (c) Segmentación obtenida. (d) Iluminación final obtenida con nuestro método. (e) Iluminación de Bousseau et al. [BPD09]. (f) Iluminación de Tappen et al. [TFA05].*



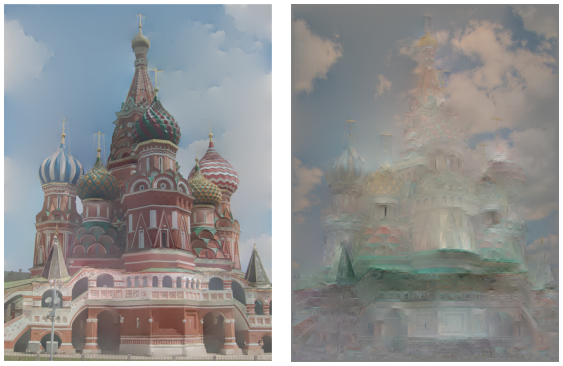
(a) Imagen original y luminancia perceptual



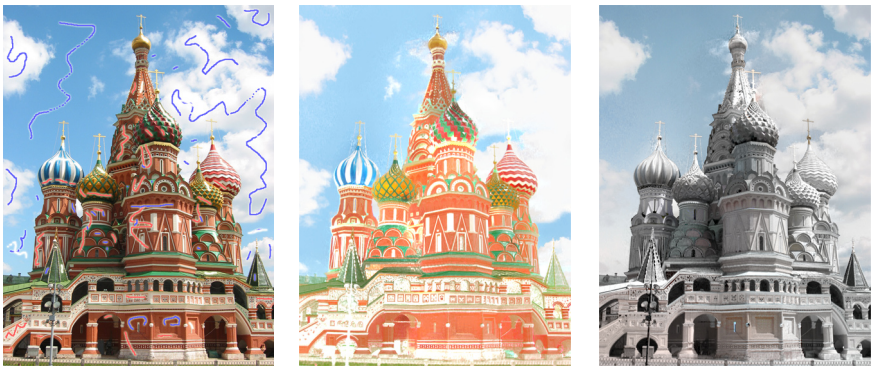
(b) Nuestra descomposición: reflectancia e iluminación



(c) Reflectancia e iluminación de Shen [STL08]



(d) Reflectancia e iluminación de Tappen [TFA05]



(e) Trazos de usuario, reflectancia e iluminación de Bousseau [BDP09]

Figura C.2: ***Comparación con otros métodos de descomposición***


(a) Imagen Original y luminancia perceptual


(e) Nuestra descomposición: reflectancia e iluminación


(c) Reflectancia e iluminación de Shen [STL08]


(b) Reflectancia e iluminación de Tappen [TFA05]



(d) Trazos del usuario, reflectancia e iluminación de Bousse [BDP09]

Figura C.3: ***Comparación con otros métodos de descomposición***

C.2. Aplicaciones

En este capítulo se muestran ejemplos de distintas aplicaciones de nuestro algoritmo de descomposición en imágenes intrínsecas.

C.2.1. Shape from Shading

Como ya se ha comentado en la introducción, el problema del *Shape from Shading* consiste en estimar la forma 3D de un objeto a partir de una sola fotografía del mismo. Debido a la ausencia de variables que acoten el problema, nos resulta imposible obtener una reconstrucción precisa del objeto. Numerosos métodos se han desarrollado y cada uno afronta el problema desde una perspectiva distinta, sin embargo, todos coinciden en realizar fuertes asunciones sobre la naturaleza de los objetos. En general, basándose en estudios de percepción, todos asumen que los objetos son globalmente convexos [LB00] y parten de la idea de *dark is deep*, es decir, cuanto más oscuro se considera más lejano y más claro más cercano. Como se puede intuir, esta última asunción, no va a dar resultados correctos cuando tratemos objetos con variaciones de albedo debidas a textura. Por este motivo, los resultados de este proyecto son una pieza clave en este tipo de algoritmos, ya que eliminamos estas variaciones de textura manteniendo las variaciones por geometría. En la Figura C.4 vemos un ejemplo del algoritmo de Shape From Shading de Wei et al. [WH97] que resultaría beneficiado por nuestra descomposición.

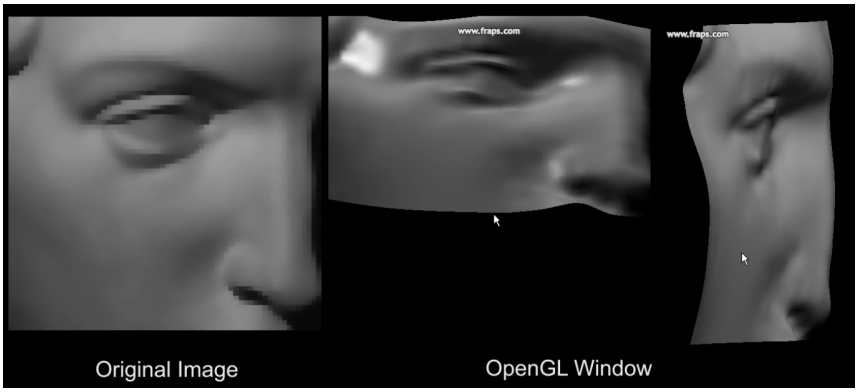


Figura C.4: ***Resultado real de Shape from Shading***. *Imagen Original y diferentes puntos de vista mostrando la reconstrucción tridimensional.*

C.2.2. Retexturización y reiluminación

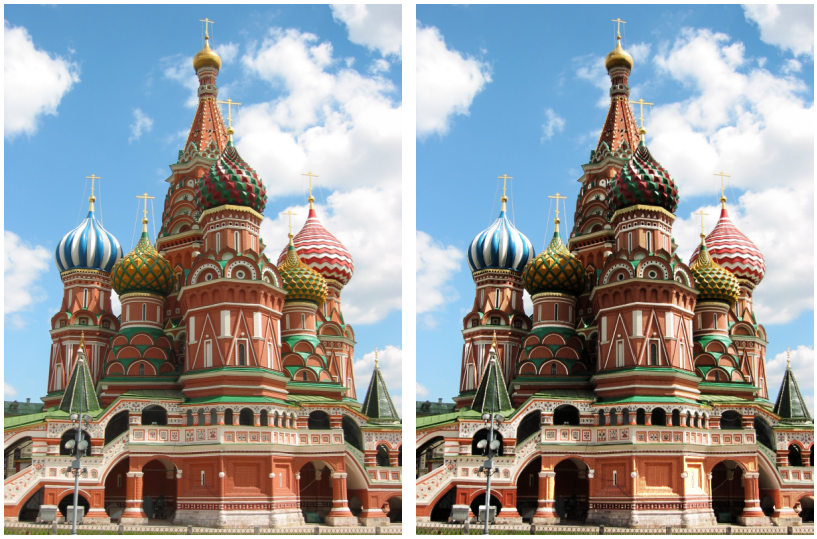
Una de las aplicaciones más básicas de la descomposición en imágenes intrínsecas es el cambio de la textura de los materiales. Una vez que hemos separado la iluminación de la reflectancia podemos estimar el mapa de normales de la imagen y aplicarlo a la imagen modificada de la reflectancia de forma similar a como lo hace Fang et al. [FH04a].

Otros algoritmos de edición de materiales a partir de una sola imagen como el de Khan et al. [KRFB06] obtendrán mejores resultados gracias a esta descomposición.

(a)

Figura C.5: *Edición de materiales de Khan et al. [KRFB06]*

Del mismo modo que podemos modificar los materiales de la imagen, tenemos la capacidad de reiluminarla de manera mucho más realista y precisa que realizar simples retoques con Photoshop. En la Figura C.6 vemos un ejemplo de esta posibilidad. Aunque para construirla nos hemos limitado a variar la componente de iluminación de una manera muy simple, podemos comprobar como se respetan las sombras producidas por la geometría y el resultado está más estilizado.



(a) (b)

Figura C.6: ***Ejemplo de reiluminación con nuestro algoritmo de descomposición.*** *(a) Imagen Original. (b) Imagen reiluminada*