ColorSide: Aplicación para cambiar los colores de objetos en imágenes de interiores.

Melanie Valverde Varas

Resumen—Este proyecto está basado en el procesamiento de imágenes, tales como: la segmentación de objetos y modificación de color. Estas funcionalidades se lograrán mediante el uso de algoritmos y herramientas que faciliten la manipulación de imágenes. En este caso, hemos optado por la librería OpenCV, la cual posee como principales ventajas: versatilidad y portabilidad. ColorSide es una aplicación de escritorio orientado al diseño de interiores, la cual dotará al usuario con la habilidad de subir la imagen a tratar, seleccionar los objetos de interés y escoger el color que desea proyectar para posteriormente mostrar la imagen con los cambios de color establecidos manteniendo la armonía de los colores originales.

Palabras Clave: Segmentación de imágenes, modificación de imágenes, OpenCV.

Abstract— This project is based on image processing, such as object segmentation and changing color. These features are achieved through the use of algorithms and tools that facilitate the manipulation of images, in this case we opted for OpenCV library, which has as principals advantages: versatility and portability. ColorSide is a desktop application-oriented interior design, which, will provide the user with the ability to upload the image to be processed, select objects of interest and choose the color you want to project to finally display the image changes color defined maintaining the harmony of the original colors.

Index Terms— Image segmentation, color modification, OpenCV.

1 Introducción

Existen diversas definiciones de visión: Según Aristóteles, "Visión es saber qué hay y dónde mediante la vista". Según Gibson, "Visión es recuperar de la información de los sentidos (vista) propiedades válidas del mundo exterior". Por lo tanto, aprovechando estas definiciones podemos decir que la visión computacional son aquellos métodos que se utilizan para adquirir, procesar, analizar y entender las imágenes adquiridas del mundo real con el propósito de crear información capaz de ser tratada por un computador.

En el siglo que nos encontramos, se podría decir que los ámbitos de diseño (gráfico o de interiores), animación, etc. Se encuentran en su máximo apogeo y es aquí donde se aplican técnicas como la modificación del color, ya sea para mejorar las propiedades de luz de una imagen, modificar las tonalidades, entre otras aplicaciones.

En este proyecto probaremos distintas técnicas, incluida la de modificación de color para crear un programa que permita a un diseñador de interiores desarrollar su creatividad dándole la posibilidad de combinar colores y objetos de un determinado espacio.

- *E-mail de contacte: immeliid@gmail.com*
- *Mención realizada: Computación.*
- *Trabajo tutorizado por: Robert Benavente i Vidal (Computer Vision Center)*
- *Curso 2015-2016*

2 Estado del Arte

En la actualidad existen diversas tecnologías que emplean la visión por computador como es el caso del reconocimiento de objetos, restauración de imágenes, análisis de video, etc. Un ejemplo claro del procesamiento de imágenes llevadas a la vida real es la medicina para el diagnóstico de enfermedades.

En cuanto a funcionalidades de nuestro proyecto, encontramos dos procesos claves a llevar a cabo: la segmentación de objetos y la transferencia de color. Hoy en día existen distintos algoritmos de segmentación como por ejemplo el algoritmo de Floodfill que permite mediante la comparación de un píxel inicial y sus respectivos píxeles vecinos, reconocer el objeto seleccionado. Una de las aplicaciones de éste algoritmo es en el juego "Buscaminas" para determinar qué piezas deben seleccionarse.

De la misma forma, entre los algoritmos de transferencia de color encontramos el propuesto por Erik Reinhard, el cual mediante el cálculo de la media y desviación estándar de lo píxeles de dos imágenes, permite trasladar las características de colores de una imagen a otra. La transferencia de colores se encuentra en la actualidad en su apogeo, sobretodo, en el área de diseño gráfico, 3D y de interiores. Existen programas online que nos permiten jugar con los colores de un ambiente del hogar, un ejemplo claro es la web interactiva de quick-step, dedicada a la producción y venta de suelos, que nos permite probar cómo quedarían en nuestro hogar los tipos de suelos laminados o de parquet que ofrecen.

3 OBJETIVOS

Durante la primera etapa de este proyecto se definieron una serie de objetivos, con la finalidad de conseguir un programa de buena calidad y por lo tanto, el éxito de este proyecto. En la siguiente tabla se expondrán los diferentes objetivos asociados a su nivel de prioridad, los cuales permitirán planificar adecuadamente las fases y tareas a seguir. Estas fases y tareas asociadas a una metodología encaminarán el desarrollo de nuestro trabajo.

Objetivos/Prioridad	Crítica	Prioritaria	Secundaria
1. El sistema debe permitir al usuario subir una imagen, seleccionar un objeto y modificar el color del mismo.		X	
2. El sistema debe procesar las imágenes de manera correcta.		X	
2.1. El sistema debe mantener los efectos de luminosidad del objeto.		X	
2.2. El sistema debe responder rápidamente a las peticiones del usuario.		X	
3. El sistema debe ofrecer una interfaz de usuario intuitiva y atractiva para una mejor interacción.			X
4. El sistema debe ser entregado en los límites establecidos.	X		

Figura 1: Tabla de objetivos asociados a su nivel de prioridad.

Objetivo 1: La funcionalidad principal de nuestra aplicación es la modificación de color de un objeto específico, por lo tanto, es esencial que el usuario pueda subir la imagen que quiera para su posterior procesamiento.

Objetivo 2: Uno de los requisitos principales respecto de la transferencia de color, es que los objetos modificados mantengan los efectos de luz y textura original. Para obtener unos resultados de buena calidad es imprescindible que nuestro programa aplique los algoritmos adecuados y que las respuestas de los mismos, sean rápidas.

Objetivo 3: Se ha propuesto realizar una interfaz atractiva e intuitiva que permita la interacción entre el usuario y el programa, sin embargo, no es un requisito obligatorio ya que su ausencia no impide las funcionalidades básicas del proyecto.

Objetivo 4: La fecha de entrega juega un papel importante ya que de ella depende el éxito del proyecto, por lo tanto, se ha calificado a este objetivo con un nivel crítico.

4 METODOLOGÍA

Se ha decidido utilizar determinados aspectos de dos metodologías que a continuación se expondrán individualmente y por último una breve explicación de las partes escogidas de cada método.

4.1 Cascada Mejorada

El propósito de éste modelo es definir el problema y las etapas que se llevarán a cabo para resolverlo. La versión original fue propuesta en el año 1970 por Winston W. Royce y posteriormente revisada por Barry Boehm e Ian Sommerville en la década de los 80. Consta en este caso de cinco fases: *Análisis de Requisitos*, se analizarán las necesidades y los objetivos del proyecto. *Diseño de Software*, desarrolla la descripción de la arquitectura del sistema de manera que se puedan llevar a cabo por separado y las herramientas que se utilizarán. *Implementación de Software*, se implementa el código fuente. *Pruebas*, los elementos programados se ensamblarán para formar el sistema y se comprobará el funcionamiento correcto del mismo. Por último tenemos a *Operación y Mantenimiento*, el software se pone en marcha, durante esta etapa pueden surgir cambios. Al finalizar cada etapa se comprobará de que estemos bien encaminados en el proyecto y se proseguirá con la siguiente fase, en el caso de detección de errores durante alguna etapa se podrá volver a la anterior, de aquí el nombre de "en Cascada mejorada". [6]

A continuación mostraremos el patrón de estructuración/organización que utilizaremos en este proyecto.

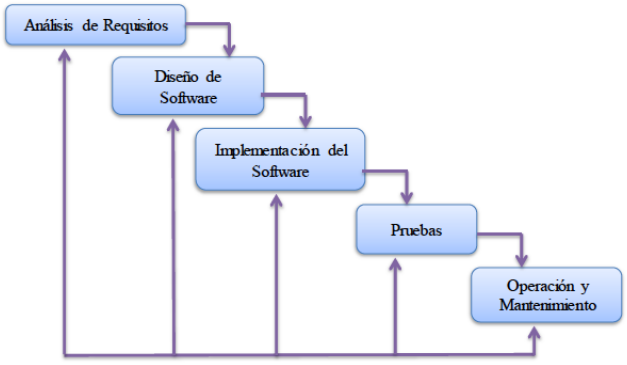


Figura 2: Modelo Cascada Mejorada.

4.2 RUP

Es un proceso desarrollado por la empresa Rational Software, en la actualidad propiedad de IBM, el cual se define como un modelo iterativo o incremental centrado en la arquitectura. La metodología RUP posee cuatro fases, cabe resaltar que por cada fase no se tiene especificado un número de iteraciones. La fase que nos interesa para nuestro proyecto es: *La elaboración*, la cual, se centra en desarrollar la línea base de la arquitectura haciendo uso de artefactos que sirven para comprender mejor el análisis y diseño del sistema (ej. Casos de uso), son estos artefactos los que utilizaremos para modelar el proyecto con la finalidad de conseguir una mejor estructuración, visualización y comprensión del sistema en cada etapa del desarrollo. [5]

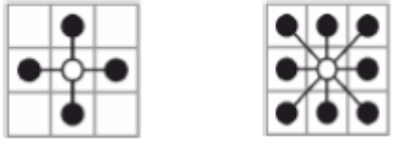
5 TÉCNICAS

En esta sección explicaremos técnicas usadas y que mejor resultados nos han proporcionado a lo largo de este proyecto.

5.1 FloodFill

Para el problema de la segmentación de objetos el algoritmo que se ha utilizado es el propuesto por S. Fazzini, llamado Floodfill (perteneciente al grupo de algoritmos Region growing), el cual nos permite obtener un área que contiene todos aquellos píxeles similares, conectados a un píxel en específico seleccionado por el usuario y perteneciente al objeto que desea manipular.

Floodfill nos permite obtener una máscara del objeto seleccionado, conectando todos los píxeles similares al pixel origen. Dentro del algoritmo podremos definir parámetros que ayuden a un cálculo más preciso, tales como los valores de tolerancia, los cuales son los rangos mínimos y máximos usados en la comparación de píxeles contiguos. Otro parámetro interesante es el tipo de conectividad, el cual, define el número de píxeles contiguos a comparar, entre los más utilizados están la conexión a 4 y la conexión a 8.



(a) (b)

Figura 3: Representación de los tipos de conectividad más utilizados. (a) 4-Conectividad y (b) 8-Conectividad.

Debemos destacar que la función original de Flood Fill nos permitiría cambiar los colores de un objeto, sin embargo, los efectos de luminosidad y textura del objeto original no se mantendrían, es por esta razón que la hemos utilizado únicamente para la parte de segmentación. Lo que se muestra a continuación son los pasos que definen al algoritmo:

1. Elegimos un píxel origen de la imagen.
2. Buscamos los 4 píxeles contiguos al píxel origen (2 horizontales y 2 verticales).
3. Calculamos la diferencia entre los píxeles vecinos respecto al píxel origen, teniendo en cuenta el rango de tolerancia definido.
4. Si los resultados de la diferencia entre píxeles se encuentran dentro de los rangos establecidos, se les asignará un color o se agregará a la máscara.
5. Sino, se termina el algoritmo.
6. Se ejecuta el algoritmo nuevamente usando el píxel izquierdo como referencia.
7. Se ejecuta el algoritmo nuevamente usando el píxel derecho como referencia.
8. Se ejecuta el algoritmo nuevamente usando el píxel superior como referencia.
9. Se ejecuta el algoritmo nuevamente usando el píxel inferior como referencia.
10. Fin del Algoritmo.

5.2 Transferencia de color

En cuanto a la modificación de color del objeto seleccionado, se ha optado por desarrollar el algoritmo de transferencia de color, propuesto por Erik Reinhard. [3] Este se basa en el uso de la media y la desviación de una imagen origen, el cual posee el color que deseamos utilizar, y una imagen objetivo, a la que aplicaremos éste cambio. Erik Reinhard expone que mediante el uso del espacio de color L*a*b*, la media y la desviación estándar de cada canal es posible trasferir el color entre dos imágenes y se centra en el espacio de color L*a*b*, debido a que en los últimos años, las investigaciones explican que este minimiza la correlación entre canales para muchas escenas naturales, lo que nos permite aplicar distintas operaciones en diferentes canales de color con cierta confianza de que no se producirán errores transversales entre canales[1]. El propósito de este procedimiento es adquirir el aspecto y la sensación de una imagen y transferirla a otra, más formalmente explicado, esto significa que nos gustaría obtener algunos aspectos de la distribución de puntos de datos en el espacio l*a*b* para transferirlas a las imágenes. Para lograr este objetivo es suficiente calculando la desviación estándar y la media de los canales de las imágenes.

A continuación se mencionarán todos los pasos del algoritmo de transferencia de color para un mejor entendimiento:

1. Coger una imagen (color) origen y una imagen destino.
2. Convertir las imágenes al espacio de color Lab.
3. Separar los tres canales: L*, a* y b* para las dos imágenes.
4. Calcular la media y desviación estándar de cada canal para cada imagen.
5. Restarle a los canales de la imagen objetivo los valores de la media de cada uno de éstos

$$\begin{aligned} l^* &= l - m(l_t) \\ a^* &= a - m(a_t) \\ b^* &= b - m(b_t) \end{aligned} \quad (1)$$

6. Multiplicar los canales de la imagen objetivo por la división entre la desviación estándar de los canales de la imagen objetivo y la desviación estándar de los canales de la imagen origen.

$$\begin{aligned} l' &= \frac{\sigma_t^l}{\sigma_s^a} \times l^* \\ a' &= \frac{\sigma_t^a}{\sigma_s^a} \times a^* \\ b' &= \frac{\sigma_t^b}{\sigma_s^b} \times b^* \end{aligned} \quad (2)$$

7. Sumarle la media de los canales de la imagen origen.

$$l'' = l' + m(l_s) \quad (3)$$
$$a'' = a' + m(a_s)$$
$$b'' = b' + m(b_s)$$

8. Combinar los canales resultantes.
9. Convertir la imagen resultante al espacio de color RGB.

6 IMPLEMENTACIÓN

A continuación aportaremos detalles de la implementación de las técnicas explicadas en la sección anterior, así como también las herramientas utilizadas en esta fase.

6.1 Herramientas de Desarrollo

En lo que respecta a herramientas de desarrollo se ha decidido hacer uso del lenguaje JAVA para la implementación del programa mediante la plataforma Eclipse.
Además se incorporará la librería OpenCV para una mayor facilidad del procesamiento de imágenes. Por otro lado, en cuanto a interfaz de usuario se ha hecho uso de la librería SWT para la creación de interfaces gráficas (GUI) con Java.

6.1.1 Eclipse

Se trata de un entorno de desarrollo integrado (IDE) de código abierto multiplataforma. Inicialmente orientado a java pero puede soportar otros lenguajes mediante plugins tales como: C/C++, PHP, Python, y plataforma Android.

6.1.2 OpenCV

Es una librería de programación libre multiplataforma (soporta varias lenguajes C/C++, java y python), está basada principalmente en el procesamiento de imágenes en tiempo real. Entre sus muchas áreas de aplicación destacarían: interacción hombre-máquina, segmentación y reconocimiento de objetos, reconocimiento de gestos, seguimiento del movimiento, estructura del movimiento y robots móviles.

6.1.3 SWT

(Estándar Widget Toolkit), es un conjunto de componentes específicos para la construcción de interfaces gráficas (GUI) en Java, desarrollados por el proyecto Eclipse. A diferencia de la biblioteca Swing, recupera la idea de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero a su vez, evita caer en las limitaciones de ésta. [4]

6.1.4 Java

Java es un lenguaje de programación orientado a objetos, creado con el objetivo de tener las mínimas dependencias de implementación. La finalidad es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo puedan ejecutar en cualquier dispositivo, es decir, el código que es ejecutado en una plataforma no tiene que ser recompilado para cada plataforma diferente.

6.1.5 Adobe Photoshop

Editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente en retoques de fotográficas y gráficos.

6.2 Segmentación de Objetos

Anteriormente se habló del algoritmo Flood Fill (perteneciente al grupo Región growing) cuyo funcionamiento se basa en un punto de referencia seleccionado por el usuario. En nuestro caso el usuario seleccionará con el mouse a través del interfaz un píxel del objeto que desea modificar. Acto seguido el programa comparará este píxel con los píxeles contiguos para así obtener la región del objeto a tratar mediante la función *Imgproc.floodfill(imagen, mask, pixel_origen, nuevo_valor, rect, umbral_min, umbral_max,flag)[3]*. Para una mejor calidad en el resultado de nuestro algoritmo, se utilizarán dos parámetros: umbral mínimo y umbral máximo, que determinan la tolerancia que se usará durante el proceso de comparación de píxeles. Usaremos el parámetro *FLOODFILL_FIXED_RANGE* para decirle al método que las comparaciones se hagan respecto al píxel actual y no entre píxeles vecinos. El tipo de conectividad asociada a nuestro algoritmo es la: 4-conexiones: solo consideramos los píxeles horizontales y verticales.

La condición de aceptación que se debe cumplir al momento de realizar la diferencia entre píxeles es la siguiente:

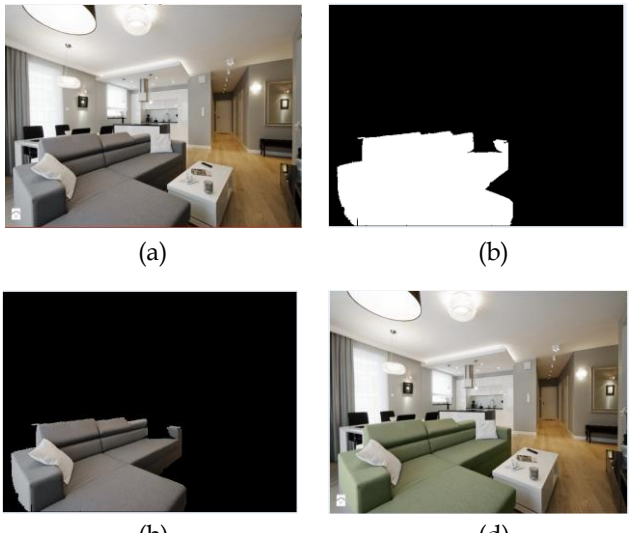
$$pixel_{semilla} - umbral_{minimo} < pixel_{contiguo} < pixel_{semilla} + umbral_{maximo}$$

6.3 Transferencia de Color

Una vez el usuario haya seleccionado el objeto que desea modificar, se creará una máscara que contendrá únicamente el objeto seleccionado. Cabe resaltar que no siempre bastará con un solo clic para que el objeto quede totalmente seleccionado, por lo que una parte de la calidad en la creación de la máscara dependerá del usuario.

En cuanto al algoritmo de Reinhard, nuestro caso es el siguiente: tenemos una máscara con el objeto que queremos tratar (obtenida del paso anterior: Flood Fill) y a la que queremos aplicar este cambio de color, por lo tanto, ésta máscara será nuestra imagen objetivo, usaremos el método *copyTo ()* para extraerla. La imagen origen contendrá simplemente el color que deseamos aplicar, es decir, su media será el valor del color en el espacio L*a*b y la desviación estándar será la misma que la de la imagen objetivo ya que nos interesa mantener la textura y efectos de luminosidad de la imagen objetivo. Lo primero que haremos será transformar las imágenes al espacio de color correspondiente, mediante la función *Imgproc.cvtColor*

(img_src, im_ dest, Color_space) [3], donde img_src es la imagen que deseamos convertir, img_dest es la imagen donde la guardaremos y Color_space es el espacio de color al cual pasaremos la imagen(BGR2Lab). A continuación, se debe separar la imagen en los 3 canales (l, a y b) mediante el uso de la función *Core.split(image, channels[])[3]*. Para cada canal se buscará la media, restaremos las medias de la imagen objetivo a cada uno de los canales l, a y b de la misma imagen. El siguiente paso es multiplicar a éstos canales la división de las desviaciones estándar de cada canal, pero como en nuestro caso son iguales, la división será 1 y no habrá ningún cambio en el resultado, es decir, este paso podría omitirse. A continuación sumaremos a cada canal las medias de la imagen origen (el color que queremos insertar). Finalmente solo nos queda plasmar la imagen del objeto con el color ya modificado en la imagen original, también haremos uso de la función *copyTo().[3]*



(a) (b)

(b) (d)

Figura 4: Orden de procesamiento de la imagen. Donde: (a) es la imagen original, (b) creación de la máscara del objeto seleccionado, (c) fusión de la imagen original y la máscara (importante para los cálculos del algoritmo de transferencia de color) y (d) Imagen final con la combinación de color deseada.

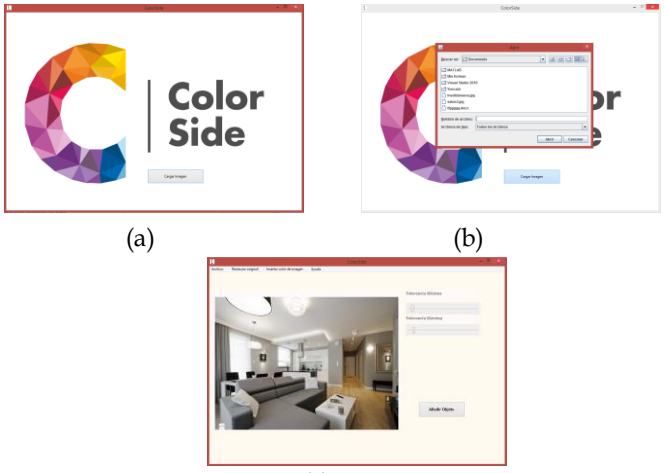
6.4 Interfaz de Usuario

Para la interfaz de usuario hemos utilizado como ya hemos comentado anteriormente SWT [8]. A continuación explicaremos las ventanas de interacción al usuario de nuestro programa.

En las dos siguientes ilustraciones podemos observar las ventanas y pasos que debe realizar el usuario para cargar una imagen:

6.4.1 Cargar Imagen

La ilustración 5(a) nos muestra la ventana principal del programa, en la que encontramos el logo del sistema, el cual ha sido diseñado mediante la herramienta Adobe Photoshop, seguido de la opción de cargar una imagen. Al presionar el botón "cargar imagen", aparecerá una ventana emergente con el directorio del usuario Figura 5(b), donde podrá elegir la imagen que desee editar. Cabe resaltar que para una mayor calidad de resultado la imagen deberá poseer una dimensión igual o mayor a 640x480p. Una vez seleccionada la imagen se mostrará la ventana observada en la Figura 5(c), en la que encontramos la imagen cargada y las diversas opciones que el usuario puede seleccionar explicadas en las secciones posteriores.



(a) (b)

(c)

Figura 5: (a) Pantalla inicial de la aplicación. (b) Opciones de cargado de imagen. (c) Imagen cargada en la aplicación.

6.4.2 Transferencia de color

Una vez el usuario haya seleccionado una imagen, podrá empezar a modificar los colores de los objetos que seleccione. El primer paso para conseguirlo será eligiendo la opción "Añadir Objeto" que aparece en la Figura 7(a), sino se elige un color antes de clicar sobre el objeto a segmentar aparecerá un mensaje de error, a continuación se mostrará el panel de colores disponibles para aplicar al objeto, el cual puede ser un color básico o un color personalizado, dependiendo de la preferencia del usuario. Después de haber escogido un color, el siguiente paso será seleccionar los valores de tolerancia. Este valor, se aplicará en el algoritmo de Flood Fill y define cuánta diferencia existirá entre dos píxeles al momento de hacer la comparación Figura 6. Es recomendable que en una imagen con objetos con mucha similitud cromática, los rangos de tolerancia sean pequeños, logrando así que nuestro algoritmo seleccione el área correcto perteneciente al objeto que queremos tratar.

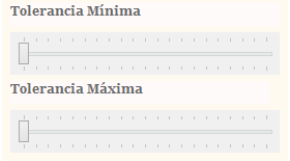


Figura 6: Valores de tolerancia.

Cuando se haya seleccionado la tolerancia que quere-

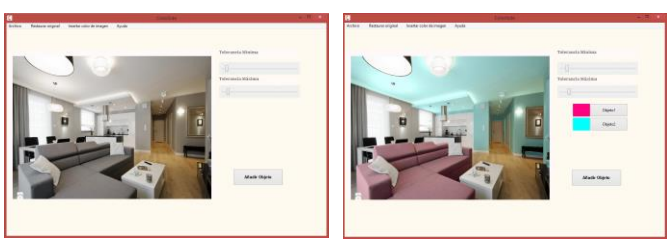
mos aplicar, el usuario deberá hacer un clic en el objeto que desea modificar, en este momento empezará a ejecutarse el algoritmo de Flood Fill. El usuario tendrá también la posibilidad de hacer click en aquellas áreas que no han sido abarcadas por nuestro algoritmo de segmentación, cada click equivale a la puesta en marcha de nuestra función de Flood Fill y la de transferencia de color. En la Figura 7(c) podemos observar el resultado final del objeto modificado.

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

Figura 7: (a) Imagen cargada. (b) Mensaje de error si no se elige un color previamente. (c) Paleta de colores. (d) Visualización del objeto modificado.

6.4.3 Multiselección de objetos

Una de las funcionalidades que se ha agregado al proyecto es la multiselección de objetos. Esta funcionalidad permite al usuario seleccionar otro objeto a modificar sin perder los cambios que haya hecho anteriormente. Los pasos a seguir son los mismos que hemos visto en la sección 6.4.2, es decir, seleccionamos la opción añadir objeto, escogemos un color y modificamos el objeto. Esta sucesión de pasos se repetirá cada vez que el usuario quiera añadir un nuevo objeto. En la figura 8(b) se puede apreciar la edición de dos objetos, en este caso, uno es el sofá y otro es la pared. Debemos resaltar que una vez se haya seleccionado la opción "añadir objeto" se guardarán los cambios que se hayan hecho en el objeto anterior y éste quedará inaccesible.



(a) (b)

Figura 8: (a) Imagen original. (b) Edición de dos objetos.

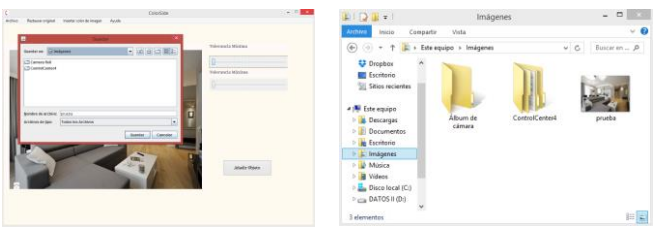
6.4.4 Menú de Ayuda

Dentro del menú de opciones que observamos en la parte superior de la aplicación, encontramos el botón de ayuda. Al hacer click a ésta alternativa se abrirá automáticamente una ventana informativa al usuario en la que se explica el funcionamiento de cada una de las opciones de la aplicación. Esto permitirá informar y resolver cualquier duda que pueda surgir durante la interacción con el programa. En la figura 9 se muestra la ventana con definiciones que nos aparecería, una vez solucionadas nuestras dudas podemos cerrarla y continuar con nuestra edición.

Figura 9: Ventana informativa del programa.

6.4.5 Restauración y Guardado de imagen.

El usuario tendrá a su disposición, las opciones de guardar la imagen editada en el path/directorio que desee, presionando el botón del menú de opciones "archivo → guarda como", a continuación escogeremos el directorio y asignaremos un nombre a nuestra imagen. Además se ha instaurado la alternativa de "restaurar original", que como bien su nombre dice, nos permitirá volver a la imagen que insertamos originalmente, borrando los cambios y objetos modificados.



(a) (b)

Figura 10: (a) Ventana de guardar imagen. (b) Imagen guardada en el directorio deseado.

6.5 Transferencia del color de una imagen a un objeto.

El usuario podrá proyectar el color global de una imagen externa a un objeto seleccionado. Para esta funcionalidad, se ha calculado los parámetros de cada canal de la imagen insertada por el usuario y en función de éstos se ha desarrollado el algoritmo de transferencia de color. El procedimiento es parecido al de la transferencia de color original, pero en este caso se utilizarán la media (medida del promedio del nivel de gris de la imagen) y desviación

estándar (medida promedio del contraste de la imagen) de la nueva imagen, es decir, el paso 6 no se omitirá ya que el valor de la división en este caso no será igual a 1, también debemos agregar que la media de la imagen origen será la media de la nueva imagen insertada ya que lo que intentamos transferir son las características cromáticas de esta.

A continuación mostramos los pasos que cambiarán respecto a nuestro algoritmo original de transferencia de color:

6. Multiplicar los canales de la imagen objetivo por la división entre la desviación estándar de los canales de la imagen objetivo y la desviación estándar de los canales de la imagen origen.

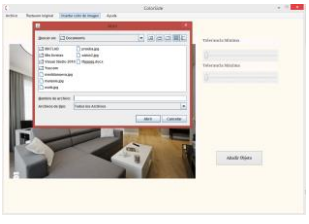
$$l' = \frac{\sigma_t^l}{\sigma_s^a} \times l^*$$
$$a' = \frac{\sigma_t^a}{\sigma_s^a} \times a^*$$
$$b' = \frac{\sigma_t^b}{\sigma_s^b} \times b^*$$

7. Sumarle la media de los canales de la imagen origen.

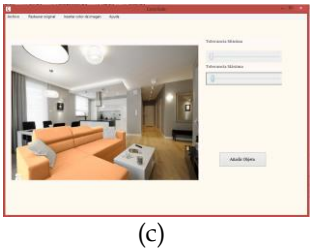
$$l'' = l' + m(l_s)$$
$$a'' = a' + m(a_s)$$
$$b'' = b' + m(b_s)$$

Esta funcionalidad la podemos activar haciendo click en la opción "insertar color imagen" en la parte superior del programa. Una vez seleccionada esta alternativa, aparecerá una ventana emergente con el directorio del usuario permitiéndole subir la imagen de la que deseamos extraer el color que posteriormente asignaremos al objeto seleccionado. Los pasos son parecidos a los que encontramos en la sección 6.4.2. Por lo tanto, Cuando hayamos seleccionado esta segunda imagen, deberemos elegir los rangos de tolerancia y acto seguido seleccionaremos el objeto al que aplicaremos la distribución de colores que posee esta nueva imagen. En las ilustraciones siguientes tenemos la imagen de la que queremos extraer la distribución de color Figura 9(a) y el resultado de la aplicación de esta distribución a un objeto en concreto en la Figura 9(c).

(a)



(b)



(c)

Figura 9: (a) Imagen a cargar. (b) Opciones de subida. (c) Resultado final obtenido.

7 Resultados

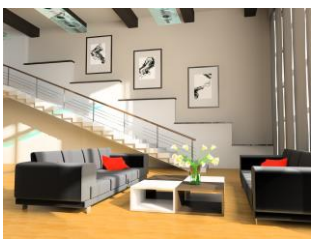
Durante la fase de pruebas nos hemos encontrado con diferentes resultados dependiendo de la imagen que procesamos, posteriormente hemos clasificado estos resultados en tres tipos de imágenes: Imágenes con superficies planas, Imágenes con superficies irregulares simples, Imágenes con superficies irregulares complejas.

A continuación explicaremos los resultados obtenidos en función del tipo de imagen tratada:

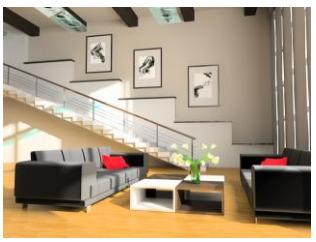
7.1 Imágenes con superficies planas

Las imágenes con superficies planas son aquellas que no poseen textura, por lo tanto la diferencia de colores entre píxeles contiguos de un mismo objeto no variará de manera desmesurada, esto tiene como consecuencia que el algoritmo de segmentación funcione de forma rápida y óptima ya que el usuario podrá seleccionar completamente un objeto haciendo el menor número de clicks sobre él, llegando muchas veces a utilizarse únicamente un click por objeto, esto depende también de los efectos de luz que exista sobre el mismo.

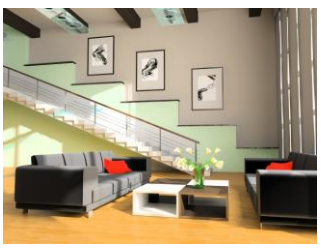
En conclusión para las imágenes planas los resultados han sido en su mayoría positivos, ya que los objetos y colores estaban bien marcados o definidos y en consecuencia, el algoritmo de transferencia de color y el algoritmo de segmentación Flood Fill funcionaba correctamente por lo que no hacía falta una gran concentración al momento de elegir el threshold o rango de tolerancia perfecto.



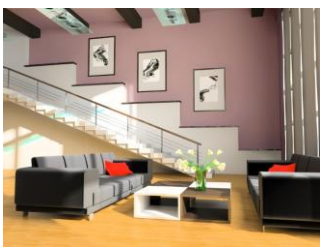
(a)



(b)



(c)

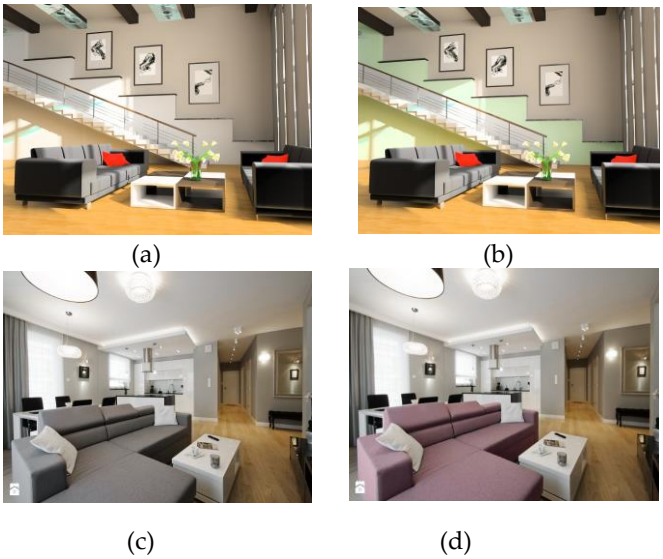


(d)

Figura 10: (a) Imagen original 1. (b) Imagen 1 con el color de los cojines modificado. (c) Imagen 1 con el pasamanos modificado. (d) Imagen 1 con la pared modificada.

7.2 Imágenes con superficies irregulares simples

Las imágenes con superficies irregulares simples son aquellas que poseen una textura constante entorno al objeto seleccionado, por lo tanto, si se escogen los rangos de tolerancia correctos se pueden segmentar de manera óptima y rápida. Por otro lado, respecto al algoritmo de transferencia de color para estas imágenes, los resultados en la mayoría de casos han sido positivos y podremos observar como la textura de los objetos y efectos de luz se mantienen.



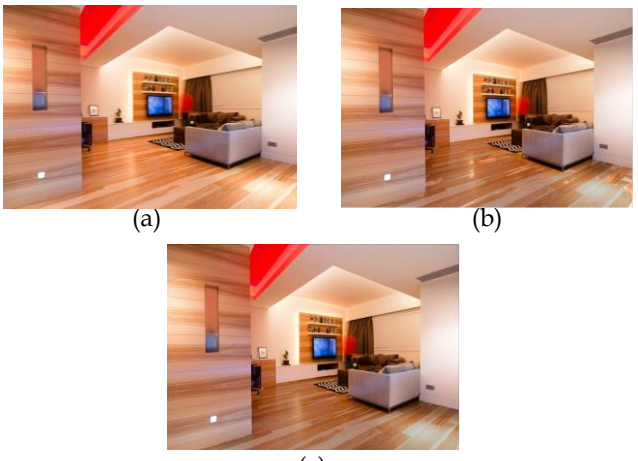
(a) (b)

(c) (d)

Figura 11: (a) Imagen original 1. (b) Imagen 1 con el pasamanos modificado. (c) Imagen original 2. (d) Imagen 2 con el sofá modificado.

7.3 Imágenes con superficies irregulares complejas

En cuanto a las imágenes de superficies irregulares complejas, son aquellas que poseen una textura en la que podrías encontrar constantemente cambios bruscos de color entre píxeles vecinos, provocando que el algoritmo de segmentación no funcione de manera óptima ya que este se basa en la comparación de píxeles contiguos y en el momento en que la diferencia entre dos píxeles se encuentre fuera del rango de tolerancia definido previamente, dará por finalizada su ejecución. Esto tiene como consecuencia que se generen trozos muy pequeños pertenecientes al objeto seleccionado y además, provocará que el usuario deba clicar un número excesivo de veces sobre las partes del objeto que no han sido abarcadas por el algoritmo. Una manera de acelerar el proceso de segmentación para este tipo de imágenes, es seleccionando rangos de tolerancia altos, pero también, cabe la posibilidad de que la máscara creada contenga píxeles no pertenecientes al objeto que queremos modificar y por ende obtener resultados indeseados. Un ejemplo de este tipo de imágenes es el siguiente: en un ambiente donde queremos seleccionar un suelo de parquet Figura 12, en el cual las irregularidades de su textura están muy marcadas con colores distintos podemos observar que es más costoso poder segmentarlo, por lo que se necesita hacer un gran número de clicks e ir jugando con los rangos de tolerancia.



(a) (b)

(c)

Figura 12: (a) Imagen original. (b) Imagen con el parquet modificado usando un numero bajo de clicks. (c) Imagen con el parquet modificado corregida usando múltiples clicks.

Las ilustraciones que aparecen a continuación reflejan los resultados de imágenes con superficies irregulares complejas (suelo) que a través de un número elevado de clicks se ha logrado segmentar por completo e imágenes con superficies planas (pared) que con solo un click han podido ser segmentadas. Volvemos a resaltar que para todos los casos los efectos de luminosidad y textura se han mantenido intactos.

(a) (b)

(c)

Figura 13: (a) Imagen original. (b) Imagen con el parquet modificado usando múltiples clicks. (c) Imagen con dos objetos modificados.

8 CONCLUSIONES

Respecto a los algoritmos utilizados, como el de Flood Fill, nos ha permitido realizar la segmentación de objetos con resultados de buena calidad, sin embargo hemos de decir que existen algunos casos en los que la segmentación de objetos es más limitada, debido a la gran similitud de distribuciones de colores entre objetos. También se debe resaltar que la interacción entre el usuario y los valores de tolerancia que elija es un factor importante para obtener éstos resultados. En cuanto al algoritmo de transferencia de color propuesta por Erik Reinhard nos permite cambiar de color el objeto seleccionado por el usuario, obteniendo buenos resultados, en los que se puede observar que se mantienen las propiedades de luz y textura de la imagen inicial.

Por otro lado, se ha añadido algunas funcionalidades extras, como el de transferencia del color global de una imagen a un objeto seleccionado, en el que a diferencia de nuestro algoritmo básico de transferencia de color, utilizará los parámetros de desviación estándar, el cual define el valor promedio del contraste de una imagen, para obtener y traspasar la distribución de colores de una imagen específica al objeto seleccionado.

Finalmente, debemos resaltar que a pesar de la poca documentación que se ha encontrado acerca de la librería OpenCV orientada al lenguaje java, se ha logrado concluir el proyecto de manera satisfactoria, cumpliendo con los objetivos establecidos al inicio del trabajo final de carrera.

9 AGRADECIMIENTOS

Me gustaría expresar mi reconocimiento a todas las personas que, de alguna forma, me han apoyado a seguir adelante y contribuido para llevar a cabo el proyecto final de carrera.

En especial a Robert Benavente, tutor de este proyecto, por su disposición a resolver cualquier duda que pudiera surgir durante el desarrollo del proyecto y por sus recomendaciones a lo largo de este tiempo.

Finalmente, también agradecer a mi familia y amigos ya que, como he mencionado anteriormente, me han brindado ánimos y soporte para continuar el proyecto y la carrera de ingeniería.

10 REFERENCIAS

[1] E. Reinhard, M. Ashikhim, B. Gooch, and P.Shirley. "*Color Transfer Between Images*", IEEE Computer Graphics and Applications, 2001.

[2] *"Algoritmo de relleno por difusión"*. [Consulta: 20/04/2016]. Disponible en: https://es.wikipedia.org/wiki/Algoritmo_de_relleno_por_difusi%C3%B3n

[3] "*OpenCV documentation"*. [Ultimo acceso: 19/05/2016]. http://docs.opencv.org/

[4] "*SWT*". [Ultimo acceso: 19/05/2016].] https://es.wikipedia.org/wiki/SWT

[5] *"Proceso Unificado de Rational"*. [Consulta: 25/02/2016]. Disponible en: https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational.

[6] "Desarrollo en Cascada". [Consulta: 25/02/2016]. Disponible en: https://es.wikipedia.org/wiki/Desarrollo_en_cascada

[7] "*Example-Based Image Manipulation*".[Ultimo acceso: 19/05/2016]. http://www.erikreinhard.com/papers/cgiv_2012.pdf

[8] Jackwind Li Guojie. *"Java Native Interfaces with SWT/JFace"*. Wiley Publishing, Inc, 2005. [Consulta: 22/05/2016]

11 ANEXO

1. Planificación del Proyecto

A continuación tenemos el diagrama de Work Breakdown Structure que tiene como objetivo presentar de manera simple y organizada el trabajo requerido para completar satisfactoriamente el proyecto.

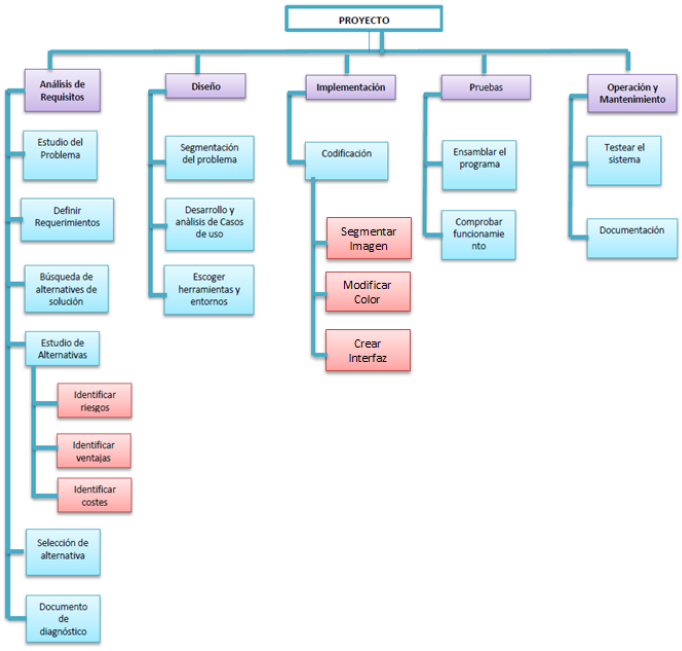


Figura 14: Diagrama Work Breakdown Structure.

En la siguiente ilustración se pueden observar las actividades de cada etapa de la organización del proyecto, como también su respectiva duración, y dependencias.

	Nombre de tarea	Durac	Comienzo	Fin	Predecesoras
1	**Interiorismo Virtual. Aplicación para cambiar los colores de objetos en imágenes de interiores.**	**93 días**	**mié 17/02/16**	**vie 24/06/16**	
2	**Análisis de Requisitos**	**16 días**	**mié 17/02/16**	**mié 09/03/16**	
3	Estudio del problema	3 días	mié 17/02/16	vie 19/02/16	
4	Definir requerimientos	3 días	lun 22/02/16	mié 24/02/16	3
5	Búsqueda de alternativas de solución	2 días	jue 25/02/16	vie 26/02/16	4
6	**Estudio de alternativas**	**3 días**	**vie 26/02/16**	**mié 02/03/16**	5
7	Indentificar riestos	3 días	lun 29/02/16	mié 02/03/16	
8	Identificar ventajas	3 días	lun 29/02/16	mié 02/03/16	
9	Identificar costes	3 días	lun 29/02/16	mié 02/03/16	
10	Selección de alternativa	2 días	jue 03/03/16	vie 04/03/16	6
11	Documento de Diagnóstico	3 días	lun 07/03/16	mié 09/03/16	10
12	**Diseño de Software**	**10 días**	**jue 10/03/16**	**mié 23/03/16**	2
13	Segmentación del problema	4 días	jue 10/03/16	mar 15/03/16	
14	Desarrollo y Análisis de Casos de Uso	5 días	mié 16/03/16	mar 22/03/16	13
15	Escoger herramientas y entornos de desarrollo a utilizar.	1 día	mié 23/03/16	mié 23/03/16	14
16	**Implementación de Software**	**44 días**	**jue 24/03/16**	**mar 24/05/16**	12
17	**Codificación**	**44 días**	**jue 24/03/16**	**mar 24/05/16**	
18	Segmentar imagen	15 días	jue 24/03/16	mié 13/04/16	
19	Modificar color	15 días	jue 14/04/16	mié 04/05/16	18
20	Crear Interfaz	14 días	jue 05/05/16	mar 24/05/16	19
21	**Pruebas**	**15 días**	**vie 20/05/16**	**vie 10/06/16**	16
22	Ensamblar todas las partes del sistema	7 días	lun 23/05/16	mar 31/05/16	
23	Comprobar funcionamiento	8 días	mié 01/06/16	vie 10/06/16	22
24	**Operación y Mantenimiento**	**10 días**	**vie 10/06/16**	**vie 24/06/16**	21
25	Testear el sistema final	8 días	vie 10/06/16	mié 22/06/16	
26	Documentación	2 días	jue 23/06/16	vie 24/06/16	25

Figura 15: Planificación de tareas y calendario.

2. **Espacio/Modelo de color:**
   Es un modelo matemático abstracto que describe la forma en la que los colores pueden representarse como tuplas de números, normalmente como tres o cuatro valores o componentes de color.
3. **Espacio de color Lab:**
   El más conocido es el espacio de color CIELAB, el cual, permite especificar estímulos de color en un espacio tridimensional. El eje *L es el de luminosidad *(lightness)* y va de 0 (negro) a 100 (blanco). Los otros dos ejes de coordenadas son a* y b*, y representan variación entre rojizo-verdoso, y amarillento-azulado, respectivamente.
4. **Flood Fill:**
   Es un algoritmo perteneciente al grupo de "región growing", que determina el área formada por elementos similares contiguos en una matriz multidimensional.
5. **Región growing:**
   Es un método de segmentación de imágenes basada en regiones o en píxeles ya que implica la selección de puntos de semillas iniciales.
6. **Segmentación de imagen:**
   En el campo de la visión artificial es el proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar.