UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

TÉCNICAS DE REUTILIZACIÓN APLICADAS A RETOQUE FOTOGRÁFICO

Autor: **José Antonio Muñoz Montero**
Director: **Profesor Dr. D. Daniel Borrajo Millán**

Año: **2011**

Agradecimientos

Gracias a Mayte y a Marcos por el apoyo que me habéis dado y por entender las ausencias.

Índice

1. *Introducción* 7

1.1 Descripción general 7

1.2 Objetivos del proyecto 8

1.3 Estructura del documento 10

2. *Estado de la cuestión* 11

2.1 Programas de manipulación de imágenes 14

2.2 GIMP 15

2.2.1 Un poco de historia de GIMP 16

2.2.2 Peculiaridades del proyecto GIMP 16

2.3 Descripción de Python 17

2.4 Cómo es un plug-in 19

2.5 Histograma 20

2.6 Distancia Euclidiana 22

3. *Objetivos del Proyecto* 23

4. *Memoria-trabajo realizado* 24

4.1 Introducción 24

4.2 Arquitectura de la aplicación 25

4.2.1 Diagrama de módulos 25

4.2.2 Diagrama de flujo de Datos 26

4.2.3 Especificación de procesos del DFD 27

4.2.4 Diccionario de datos 29

4.3 Descripción a alto nivel de cada modulo 30

4.3.1 Estructura de un plug-in 30

4.3.2 Procedimientos 35

4.3.3 INICIO.py 36
4.3.4 FIN.py 39
4.3.5 MACROS.py 41
4.3.6 RECOMENDACIÓN.py 42
4.4 Manual de usuario 48
4.4.1 Uso de Procedimientos 50
4.4.2 Grabar una nueva macro 56
4.4.3 Ejecutar una macro 57
4.4.4 Recomendación automática de macro 57
4.5 Manual de referencia 60
5. *Resultados* 62
5.1 Pruebas de validación 63
5.2 Prueba del cálculo de la distancia euclidiana. 66
5.3 Pruebas de funcionamiento 67
6. *Gestión del proyecto* 71
7. *Conclusiones* 73
8. *Futuras líneas de trabajo* 75
9. *Bibliografía* 77
10. *Anexos* 78
10.1 Instalación y configuración del sistema 78
10.2 Ficheros de la aplicación 80

Índice de Figuras

Figura 1. Estructura básica de un *plug-in* o complemento. 19
Figura 2. Presentación del Histograma de una imagen en GIMP. 21
Figura 3. Diagrama de Módulos 25
Figura 4. DFD Diagrama de flujo de datos..... 26
Figura 5. Visor de procedimientos de GIMP 32
Figura 6. Menú `<Image>` de GIMP..... 33
Figura 7. Cuadro de dialogo. 34
Figura 8. Rangos de los niveles de luminosidad. 38
Figura 9. Imagen "`palomas.jpg`" editada con GIMP..... 38
Figura 10. Histograma en GIMP de la imagen "`palomas.jpg`" 39
Figura 11. Cuadro de dialogo del *plug-in* "`fin.py`" 40
Figura 12. Cuadro de dialogo del *plug-in* "`macros.py`" 41
Figura 13. Cuadro de dialogo del *plug-in* "`recomendacion.py`" 43
Figura 14. Diagrama de flujo de la selección de modo. 46
Figura 15. Menú de la aplicación..... 49
Figura 16. Imagen "`motos.jpg`" antes de aplicar "`Desaturar`" 51
Figura 17. Cuadro de Diálogo del procedimiento "`Desaturar`"..... 51
Figura 18. Imagen "`motos.jpg`" después de aplicar "`desaturar`". . 52
Figura 19. Antes y después de aplicar "`filtro NL`"..... 53
Figura 20. Cuadro de diálogo del procedimiento "`pixelar`". 54
Figura 21. Antes y después de aplicar "`pixelar`". 54
Figura 22. Efecto conseguido al aplicar "`rotar`". 55
Figura 23. Cuadro de dialogo del *plug-in* "`recomendacion.py`" 58
Figura 24. Mensaje de recomendaciones..... 59
Figura 25. Petición de recomendación en modo `K-minimos`. 67
Figura 26. Mensaje de recomendación y la imagen retocada. 68
Figura 27. Imagen "`Pirámides.BMP`"..... 68
Figura 28. Recomendaciones en modo "`dispersión`" y K=3. 69

Figura 29. Recomendaciones en modo "aleatorio" y K=3..... 69
Figura 30. Imagen original (izquierda), y la imagen retocada. 70

1. *Introducción*

1.1 *Descripción general*

Este proyecto está relacionado con el tratamiento informático de la fotografía digital. Es uno de los campos donde más ha evolucionado la informática. Se trata de todo lo relacionado con la manipulación y edición de fotografía digital. La informática actual ofrece una gran variedad de programas relacionados con la fotografía digital.

El software de retoque fotográfico es usado para mejorar la calidad de imágenes y eliminar defectos de las mismas, pero también para crear nuevas imágenes o incluso para hacer transformaciones artísticas de todo tipo de imágenes. Y todas estas características están al alcance de casi cualquier tipo de usuario, desde los profesionales que cuentan con una herramienta de edición que les ayuda a mejorar y facilitar su trabajo, hasta los usuarios particulares que gracias a la expansión de la fotografía digital pueden hacer uso de estas herramientas para manipular fotografías personales.

Existen multitud de programas de manipulación de fotografía digital, pero hay uno que podemos usar como referente para explicar a todo tipo de público a qué tipo de software nos estamos refiriendo, y es el *Photoshop*. Es muy fácil conseguir explicar cuál es el ámbito de trabajo de este proyecto, incluso a personas totalmente ajenas al mundo de la fotografía y de la informática, si se usa como referencia el nombre de este conocido programa.

Este proyecto, ha añadido a las ya existentes una serie de funcionalidades dirigidas a facilitar la labor del retoque fotográfico. Se ha elegido el programa GIMP (GNU[1] *Image Manipulation Program*) un programa de gran reputación en el mundo del *software* libre. Es un

[1] GNU es un acrónimo recursivo que significa GNU no es Unix.

programa usado en tareas de retoque fotográfico y composición y edición de imagen. Es completamente extensible, es decir, se le pueden incorporar todo tipo de extensiones y módulos para aumentar su funcionalidad.

La condición de altamente extensible de GIMP que se ha mencionado ya permite elegir entre un variado número de lenguajes de programación. Tras un periodo de estudio del programa GIMP, se ha decidido utilizar *Python*[2] para crear el código. Este lenguaje está perfectamente integrado con GIMP, ofreciendo librerías que facilitan el trabajo. *Python* es un lenguaje interpretado o de *script*[3] que se ha convertido en la sorpresa agradable del trabajo de este proyecto.

Un aspecto importante tanto de GIMP como de *Python* es que son multiplataforma, lo cual garantiza que el resultado obtenido funcione en los principales sistemas operativos. Así pues el trabajo de diseño, desarrollo, depuración y pruebas del *software* se ha realizado en *Windows*, pero también se ha comprobado su funcionamiento en Mac OS X[4] (*Macintosh Operating System*).

1.2 *Objetivos del proyecto*

Este proyecto pretende aportar una nueva forma de usar funcionalidades ya conocidas de los programas de retoque fotográfico. La idea consiste en usar la relación que hay entre las características de una fotografía y la funcionalidad que se le aplica, de manera que se pueda llegar a recomendar qué funcionalidades usar con una fotografía dependiendo de sus características.

[2] Python es un lenguaje interpretado de alto nivel.
[3] Un script es un fichero que contiene un conjunto de instrucciones que deben ser interpretadas línea a línea en tiempo de ejecución.
[4] X se refiere al diez en números romanos.

A veces el trabajo de edición para retocar una imagen no consiste únicamente en aplicar una determinada funcionalidad, si no que puede requerir la combinación de varias de ellas, de manera que puede ser interesante y útil encontrar la manera de volver a usar esa misma combinación, incluyendo los valores exactos de los parámetros usados. Por eso, este trabajo ha estado dirigido a conseguir tener el control de las funcionalidades a un nivel que nos permita guardarlas con sus parámetros incluidos, bien individualmente o en combinaciones. Una vez conseguido este control, se trata de poder usarlo, es decir, ejecutar estas funcionalidades que hemos guardado cada vez que se quiera. Y por último, una vez que se tiene el control de dichas funcionalidades, el objetivo es recomendar al usuario la funcionalidad que puede aplicar a una nueva fotografía o imagen.

Centrándonos en GIMP, cuando lo usamos podemos aplicar una de sus herramientas o de sus filtros a una fotografía, y si nos parece bien podemos guardar la imagen retocada pero no la acción o acciones que hemos aplicado. Este trabajo pretende también almacenar la acción o las acciones que se han aplicado sobre ella. Un ejemplo sencillo, podría ser el caso de tener una foto a la que le aplicamos un cambio en el brillo, y un retoque en el balance de color. Una vez terminado el trabajo podemos tener almacenado una "nueva función" con los parámetros que hemos usado, que haría lo mismo sobre la fotografía que queramos. Una vez hecho esto lo que conseguimos es una serie de funcionalidades, que están construidas con las funcionalidades originales del programa, pero que podremos usar de forma diferente. Según se va usando se van acumulando más funcionalidades de todo tipo, junto con los parámetros de las imágenes con las que han sido usadas. Y toda esta información es la que usará el programa para hacer recomendaciones al usuario. Así pues la intención es conseguir ayudar a un usuario a elegir qué

funcionalidad puede requerir su imagen, en función de las características de la propia imagen.

1.3 *Estructura del documento*

En el capítulo 2 se habla del retoque fotográfico por ordenador, como campo en el que se enmarca este proyecto. Para centrar el tema de trabajo se habla de los programas de manipulación de imágenes. Luego se habla extensamente del programa en el que se desarrolla el proyecto así como del resto de elementos usados, el lenguaje, la herramienta histograma y de la distancia euclidiana.

En el capítulo 3 se presentan los objetivos del proyecto.

En el capítulo 4 se dedica a la explicación del trabajo realizado, la arquitectura de la aplicación, una descripción a alto nivel de todos los módulos y manuales de usuario y de referencia.

En el capítulo 5 se habla de los resultados obtenidos con las pruebas realizadas a la aplicación.

En el capítulo 6 se da información referente a la parte de gestión del trabajo realizado.

El capítulo 7 presenta brevemente las conclusiones obtenidas en la elaboración del proyecto.

En el capítulo 8 se han expuesto algunas ideas y sugerencias sobre las posibles líneas de trabajo a seguir.

El capítulo 9 muestra la bibliografía utilizada en este documento.

Y en el capítulo 10 de anexos se encuentra el código de la aplicación.

2. *Estado de la cuestión*

Seguramente haya muchos programas de manipulación y edición fotográfica que hagan cosas similares a las buscadas por este proyecto. Tras examinar varios de ellos, aquí solo se va a hacer referencia a uno de ellos como muestra práctica y ayuda para la explicación del trabajo a desarrollar. Se trata del programa *Jasc Paint Shop Pro* (http://www.corel.com) que tiene una función muy parecida a la buscada por este proyecto. Esta función permite grabar una macro para automatizar tareas. El funcionamiento es sencillo: se inicia la grabación de un *script,* y mientras está activada la grabación se guardan todas las tareas que se van haciendo. Cuando se finaliza la grabación, se cierra un *script* que contiene todas las instrucciones que representan las tareas realizadas con los parámetros usados. Después se puede ejecutar el *script* con lo cual se repiten las tareas automáticamente. El *script* se guarda en un formato propio con el nombre y ubicación elegidos por el usuario. Los *script* generados son ficheros independientes y accesibles mediante una herramienta de edición del propio programa o desde cualquier editor para poder modificarlos. Este precedente es muy interesante, ya que implementa parte de la funcionalidad que se quiere conseguir, y esta es la idea elegida para este proyecto.

Otra parte no menos importante del proyecto pretende conseguir un mecanismo capaz de ofrecer al usuario automáticamente la recomendación de que acciones aplicar a la imagen, basándose en la experiencia de casos anteriores. Actualmente existen técnicas de Razonamiento Basado en Casos, que resuelven problemas similares a los planteados en este proyecto. El razonamiento basado en casos se puede definir como un proceso que consta de cuatro pasos: Recuperar, Reutilizar, Revisar y Guardar. En este trabajo no se han

estudiado estas técnicas ni se sigue estrictamente la forma de trabajar, aunque sí que guardan un parecido.

En este proyecto se parte de la base de trabajar a partir del programa de manipulación de imágenes GIMP. Para iniciarse en GIMP es muy recomendable visitar la página oficial de GIMP http://www.gimp.org en la que están disponibles para descargar legalmente las versiones de instalación en muchos idiomas para los sistemas operativos más importantes. También se pueden encontrar manuales de uso de la aplicación completa, noticias sobre nuevas versiones, información de las antiguas, documentación de todo tipo, el código fuente del GIMP, información para desarrolladores, y prácticamente todo lo relacionado con GIMP. Otra buena recomendación complementaria a la anterior es http://www.gimp.org.es/ en la que se marcan como objetivo acercar el GIMP a todos los hispanohablantes. En ambos sitios se puede acceder a distintos foros y listas de correo en las que se puede indistintamente resolver dudas o colaborar activamente. Además de estas páginas, es fácil encontrar en internet ayuda de todo tipo para seguir profundizando en las posibilidades que ofrece GIMP para ampliar su funcionalidad a base de módulos y extensiones. Existen muchas páginas y foros de aficionados en los que se puede encontrar las instrucciones para instalar y configurar un sistema en el que poder crear ampliaciones para GIMP.

Después del acercamiento inicial a GIMP, se trata de elegir la forma en la que se va a trabajar con GIMP; es decir, qué lenguaje y herramientas usar para implementar el código. Como ya se ha dicho hay muchas formas de añadirle funcionalidades a GIMP. Aquí no se van a abordar todas, pero se van a mencionar algunas de ellas, que han sido estudiadas, aunque por unos u otros motivos han sido descartadas para este proyecto. Una de las opciones para trabajar en GIMP es *Script-fu*, que es lo que en el mundo *Windows* se

denominaría "macros", aunque es mucho más potente, y está basado en *Scheme*[5]. También se contempla la posibilidad de estudiar el código fuente de GIMP que está disponible en C++ y sobre él hacer modificaciones y ampliaciones.

El lenguaje elegido finalmente para hacer todo el desarrollo de la aplicación es *Python*. Este es un lenguaje altamente integrado en GIMP, y permite usar los complementos y el interface gráfico de GIMP.

Una vez elegida la plataforma (*Windows*), la aplicación principal (GIMP) y lenguaje con el que trabajar (*Python*), es necesario estudiar la forma en la que interactuar con GIMP. Se han estudiado varias formas de interactuar con GIMP antes de llegar a una conclusión definitiva, y aunque no han sido usadas, algunas de estas opciones van a ser mencionados aquí.

Se ha estudiado el formato XMP[6] que es un tipo de lenguaje especificado extensible usado en archivos PDF[7] y en archivos de fotografía. Este formato permite definir un modelo de metadatos en el que tener la información deseada y almacenarla asociada al propio archivo de la fotografía. GIMP permite trabajar con estos metadatos que pueden contener información de todo tipo; desde la fecha de la foto, hasta los ajustes de la cámara para realizar la foto pasando por cualquier tipo de datos que se quieran guardar.

También se ha estudiado la posibilidad de utilizar la herramienta ya existente en GIMP del "Historial de deshacer" que permite deshacer los cambios que se hacen en una imagen. Esta herramienta está basada en dos pilas de datos, una llamada UNDO y la otra REDO, que almacenan las imágenes resultantes de aplicar sobre ellas las

[5] Scheme es un lenguaje interpretado.
[6] XMP Extenseble Metadata Platform o Plataforma Extensible de metadatos.
[7] PDF Portable Document Format o Formato de Documento Portable.

diferentes acciones o herramientas de GIMP. El usuario puede coger las imágenes de una u otra pila, teniendo de esta manera la secuencia de imágenes con las modificaciones hechas, y quedándose con la que más le interese. Pero esta herramienta no se puede manejar mediante procedimientos de GIMP, y además lo que guarda son imágenes, y no las acciones que se aplican a las imágenes.

A continuación se va a profundizar sobre los aspectos que han aparecido hasta ahora en este documento.

2.1 *Programas de manipulación de imágenes*

El mundo de la fotografía siempre ha contado con aficionados, a la vez que es una profesión con una connotación casi artística. El retoque y la edición han sido actividades muy restringidas solamente a los ambientes profesionales.

Sony presento su primera cámara digital en 1989, y en muy poco tiempo el retoque y la edición de imágenes ha crecido y se ha perfeccionado de tal forma, que se ha hecho muy accesible a todo tipo de usuarios, y no solo a los profesionales.

Este avance ha sido provocado por varios factores, y entre ellos está el avance de la informática, por un lado con la mejora del rendimiento del *hardware* que ha elevado la capacidad de proceso disponible, que por otro lado el *software* a conseguido rentabilizar y maximizar. Y es aquí donde entran los programas *software* de retoque y edición de fotografías, que es el entorno de trabajo de este proyecto.

Hoy día existe una gran variedad de programas informáticos que permiten la manipulación de imágenes, incluso han alcanzado un nivel de perfeccionamiento espectacular, de forma que ha conseguido

que popularmente se conozcan algunos de ellos gracias a manipulaciones que en momentos determinados han llamado la atención de la opinión pública. *PhotoShop* se ha convertido en un *software* conocido incluso por gente que no ha sacado una foto en su vida ni ha usado un ordenador nunca.

Existen pues multitud de programas informáticos dedicados al retoque y edición de imágenes. También existen gran cantidad de formatos de ficheros de imágenes aunque aquí solo se ha trabajado con GIMP y el formato de fichero de imagen que se ha usado es el JPEG[8] (*Joint Photographic Experts Group*), aunque como ya se puede suponer, tanto GIMP como el resto de programas son capaces de trabajar con muchos más formatos.

2.2 *GIMP*

GIMP es un programa libre apropiado para tareas como retoque fotográfico y composición y edición de imagen.

GIMP es completamente extensible, lo que quiere decir que ha sido diseñado para que puedan incorporarse extensiones y *plug-ins*[9] que aumenten su funcionalidad. Su interfaz de *script* permite que para cualquier tarea se pueda crear un *script* y automatizarla.

Aunque está escrito y desarrollado bajo X11 en plataforma *Unix*, básicamente el mismo código también funciona en *Microsoft Windows* y *MAC OS X*.

[8] JPEG Joint Photographic Experts Group o Grupo Conjunto de Expertos en Fotografía. Es uno de los formatos más extendidos para los ficheros de fotografía.
[9] Plug-in o complemento es una aplicación que se relaciona con otra para aportar alguna función nueva.

2.2.1 Un poco de historia de GIMP

El proyecto GIMP fue iniciado por un par de estudiantes de *Berkeley*, *Peter Mattis* y *Spencer Kimball* en 1995. Protegieron la primera versión del programa resultante con la licencia GPL (*General Public License*) y tuvo bastante éxito. Se creó una lista de correo para desarrolladores, que es la forma en la que aun hoy sigue creciendo la aplicación. Empezaron a surgir sitios *web*, a parte del sitio principal de *Berkeley*, que se dedicaban a explicar cómo se usaba GIMP. Se escribían manuales de usuario y no dejaban de crearse obras con GIMP (por ejemplo *Larry Ewing* creó el famoso pingüino *Tux* con GIMP 0.54, http://www.isc.yamau.edu/~lewing/linux/).

Aunque sus creadores abandonaron el proyecto al graduarse, nuevos desarrolladores dieron un paso al frente y siguieron trabajando en el proyecto, haciendo que hoy día GIMP sea uno de los mejores y más populares programas del mundo del *software* libre.

2.2.2 Peculiaridades del proyecto GIMP

Aunque no es el objeto de este proyecto, no está de más conocer algunos detalles acerca de cómo puede funcionar en nuestros tiempos un proyecto basado en una estructura de desarrollo anárquica, en la que algunos de los desarrolladores no tienen motivación económica alguna. Del documento "El proyecto GIMP" (León, 2008), en el que se explica bastante bien el inicio y funcionamiento del proyecto GIMP, se sacan algunas ideas generales dignas de ser comentadas. El equipo de desarrollo de GIMP está organizado a partir de una lista de correo abierta a la que cualquiera puede pertenecer. De hecho en la fase de estudio de GIMP necesaria para acometer este proyecto, yo mismo he podido acceder a la lista de correo de desarrolladores donde he expuesto mis dudas, que han sido atendidas y resueltas

suficientemente. Esto demuestra la utilidad y eficacia del método de desarrollo de GIMP.

En un análisis de las direcciones de correo electrónico de los colaboradores que forman parte de la lista de correo de desarrollo, se observa que hay una división en varios grupos. Hay desarrolladores que pertenecen a universidades y a centros de desarrollo o a empresas que colaboran de alguna manera. Pero el grupo más interesante es el de los colaboradores que usan una dirección de correo personal, lo que es interpretado como una colaboración altruista. Dichos desarrolladores hablan de las motivaciones que tienen para colaborar en el proyecto GIMP en su tiempo libre. Son tan simples como el hecho de que disfrutan haciéndolo, o por compartir las ideas de libertad que propone un sistema de software libre. El documento mencionado, muestra una serie de comentarios y anécdotas que ponen de manifiesto estas ideas tan peculiares.

No se trata de presentar este sistema de desarrollo como un modelo de perfección, ya que a las idílicas ideas expuestas anteriormente, se suman características negativas como las dificultades en la toma de decisiones o en el gobierno de la estructura de desarrollo.

2.3 *Descripción de Python*

Python es un leguaje de programación creado por Guido Van Rossum a principios de los años 90.

Es un lenguaje interpretado o de *script*; es decir, se ejecuta por medio de un intérprete y no necesita ser compilado. Esta característica es el punto débil desde el punto de vista de la depuración de los programas con *Python*. Especialmente cuando se empieza, ya que resulta difícil encontrar los errores que se cometen en el código. Otra desventaja es que es más lenta la ejecución, pero esto se compensa por la ventaja que da su flexibilidad y portabilidad.

Python es un lenguaje de tipado dinámico; es decir, no es necesario declarar el tipo de dato que va a contener una variable sino que su tipo se determinará en tiempo de ejecución según el tipo de valor que se le asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Otra característica es que es fuertemente tipado; no se permite tratar a una variable como si fuera de un tipo distinto al que tiene. Es necesario convertir de forma explícita dicha variable al nuevo tipo previamente.

Para explicar estas dos características que pueden resultar contradictorias, se toma el ejemplo de la variable `"pi"`. El tipado dinámico permite que con solo escribir "`pi = "3.1416" "` la variable queda definida como de tipo `"string"`. Pero si se necesita usar el valor de `"pi"` en la operación `"L = 2 * pi * r"`, se comete un error de sintaxis, por que debido a la característica de fuertemente tipado del lenguaje "pi" no tiene un valor numérico, entonces hay que explicitar el cambio de tipo: `"L = 2 * float(pi) * r"`. Otra forma de cambiar el tipo de `"pi"`, es directamente asignándole un valor numérico: `"pi=3.1416"`.

El intérprete de *Python* está disponible en multitud de plataformas, por lo que si no utilizamos librerías especificas de cada plataforma nuestro programa podrá correr en todos los sistemas sin cambios. La aplicación creada en este proyecto funciona correctamente en *Windows* (varias versiones) donde se ha hecho el desarrollo y depuración del código, y en MAC OS X donde se ha comprobado el funcionamiento.

Python también permite varios paradigmas de programación como la programación imperativa, funcional y orientada a aspectos, y por su puesto orientada a objetos. Una de las características de *Python* es

que todo son objetos, y como tales se les puede tratar, aunque no se declaren explícitamente. Por ejemplo, una cadena de caracteres es tratada como un objeto al que se le pueden aplicar métodos muy potentes pensados para listas.

2.4 *Cómo es un plug-in*

Todos los complementos comparten una estructura básica común, que además debe ser respetada para su buen funcionamiento. Cada *plug-in* o complemento está contenido en un fichero con la extensión "py".

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python
# Se importan los módulos necesarios
from gimpfu import *
import os

# definimos las funciones necesarias
def desaturar(img, drawable,modo):
    pdb.gimp_desaturate_full(drawable,modo)#Ejecucion del procedimiento desaturate
    try:
        file = open("archivo.tmp.xml", "r")
    except:
        pdb.gimp_message("No se ha iniciado proceso.No se guardara esta accion.")
    else:
# función principal
if __name__ == '__main__':

    # llamada a función register
    register(
        "desaturar",
        "Convertir los colores en niveles de gris",
        "Convertir los colores en niveles de gris",
        "Jose Antonio",
        "Jose Antonio",
        "2011",
        "<Image>/Gestion de Tareas/Procedimientos/Desaturar",
        "RGB*, GRAY*",
        [
            (PF_RADIO, "modo", "Seleccionar : ", "0", (("Claridad", "0"))
        ],
        [],
        desaturar)
    main()
```

Figura 1. Estructura básica de un *plug-in* o complemento.

La Figura 1 muestra un ejemplo de la estructura básica de un *plug-in.* Este es el contenido del fichero "`desaturar.py`", y este código implementa la función desaturar, que consisten en quitar el color de la imagen sobre la que se aplica.

2.5 *Histograma*

El histograma es una de las herramientas gráficas más útiles para resumir los valores de una variable referidos a su conjunto de datos. Un histograma es la representación gráfica de una tabla de frecuencias donde los datos han sido agrupados por intervalos.

Aplicado a la fotografía, el histograma representa los distintos niveles de luminosidad recogidos en la imagen. Hoy día prácticamente cualquier cámara cuenta con la representación del histograma como una opción de visualización de la imagen. Así pues es una herramienta muy útil para mejorar la calidad de las fotografías. Una de las técnicas más básicas de retoque fotográfico es la modificación del histograma de la imagen. Además, puede usarse como una herramienta infalible para comprobar si la imagen que se quiere fotografiar está correctamente expuesta. De esta forma desaparece la sorpresa al ver una foto en el ordenador que sale demasiado oscura, y que en el momento de hacerla se veía bien en la pantalla de la cámara. La imagen de la foto que vemos en la pantalla de la cámara puede ser engañosa por las condiciones de la luz. Sin embargo, haciendo una correcta representación del histograma se garantiza que la foto está expuesta correctamente. El histograma presenta en la parte izquierda los tonos más oscuros, y a la derecha los más claros. Por lo tanto un histograma con los valores de la parte izquierda más altos es propio de una imagen subexpuesta, es decir, demasiado oscura. Y al contrario estaremos hablando de una imagen

sobreexpuesta o quemada. Queda claro que lo que se debe buscar es un histograma con valores repartidos.

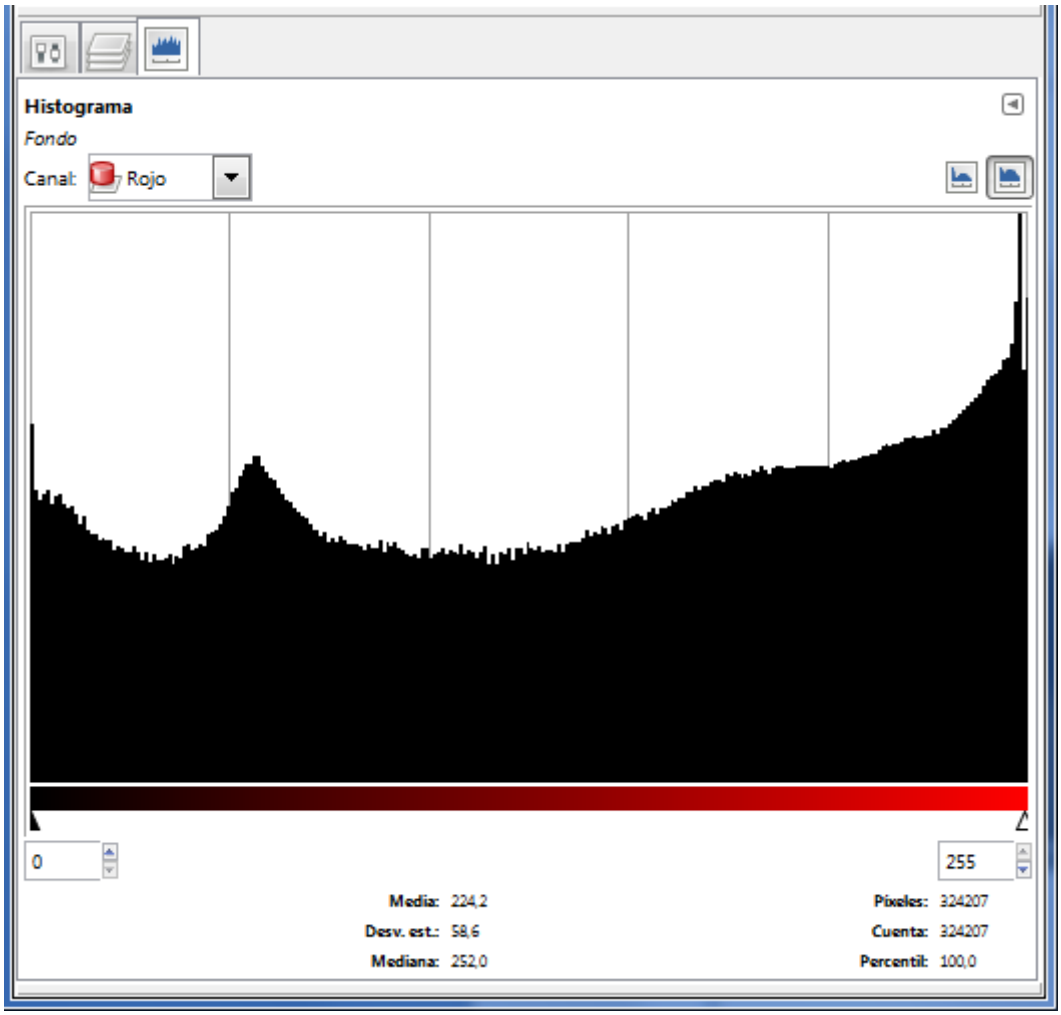

Figura 2. Presentación del Histograma de una imagen en GIMP.

Gimp, al igual que otros muchos programas de retoque fotográfico permite obtener el histograma de la imagen con la que se está trabajando. El grafico que GIMP muestra, representa la cantidad de píxeles que hay en cada nivel de luminosidad de hasta 6 canales diferentes. En el eje horizontal tenemos los 256 niveles de luminosidad por cada canal. En el eje vertical la cantidad de píxeles de cada nivel.

Además del gráfico, GIMP presenta una serie de datos estadísticos: la media aritmética; la desviación estándar; la mediana; número total

de píxeles que tiene la imagen en el canal; la cuenta, que es el numero de píxeles que hay en el rango elegido; y el percentil, que es el porcentaje del total que representan los píxeles del rango elegido.

2.6 *Distancia Euclidiana*

La distancia euclidiana es la distancia entre dos puntos. Los puntos pueden pertenecer a un espacio de dos o más dimensiones, así pues la definición de un punto vendrá dada por dos o más coordenadas. La distancia entre el punto A definido por las coordenadas $(a_1, a_2, \dots, a_n)$ y el punto B definido por $(b_1, b_2, \dots, b_n)$ es

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

En este proyecto los puntos van a ser imágenes definidas mediante una serie de valores numéricos de sus histogramas. Se calcula la distancia euclidiana entre dichas imágenes para determinar que las que tengan entre si la mínima distancia euclidiana, serán las más similares.

3. *Objetivos del Proyecto*

El primer objetivo que persigue el proyecto es el de crear una herramienta que permita manejar macros en GIMP. Esta herramienta debe ofrecer la posibilidad de guardar en un formato determinado las acciones que se realicen sobre una determinada imagen. Las macros resultantes deben poder volverse a ejecutar sobre cualquier otra imagen, o bien ser editadas y modificadas. Así se puede obtener una variada colección de macros, que contienen la acción o acciones que se han usado para una determinada imagen. Esta colección de macros se registra junto con la información de la imagen a la que ha sido aplicada. Y de aquí se deriva el segundo objetivo del proyecto, que es el de implementar un mecanismo de razonamiento basado en casos para facilitar la labor de los usuarios a la hora de tratar imágenes futuras. Para ello, la herramienta debe poder sugerir la utilización de diversas macros almacenadas previamente. Lo hará en función de la semejanza de las nuevas imágenes con las tratadas previamente.

4. *Memoria-trabajo realizado*

4.1 *Introducción*

La aplicación creada está dividida en *plug-ins* o complementos que tienen una estructura básica idéntica. Estos complementos son ejecutados por la aplicación principal (GIMP) e interactúan por medio del API[10]. Esta forma elegida para trabajar permite incorporar en cualquier momento más complementos para ampliar o mejorar la funcionalidad de la aplicación.

La forma elegida para interactuar con GIMP es guardar en ficheros externos las acciones que se realizan sobre una foto, de forma muy similar a como lo hace *Jasc Paint Shop Pro* (http://www.corel.com). Cuando se ejecutan los procedimientos de GIMP, se almacena en un fichero el nombre de esos procedimientos y de los parámetros que se han usado. Los ficheros que se guardan, similares a lo que llamamos macros, se crean respetando el formato XML[11], para seguir un estándar. Estos ficheros podrán ser ejecutados de nuevo cuando se desee y conseguirán sobre la imagen que se apliquen el mismo efecto que cuando se grabaron. La secuencia desde el punto de vista del usuario es iniciar grabación de una macro, aplicar los procedimientos deseados sobre la imagen y finalizar la grabación de la macro.

Además de generar un fichero XML o macro, la aplicación simultáneamente, almacena los datos relativos a los retoques hechos a cada imagen. Concretamente se almacenan el nombre de la imagen retocada, los valores del histograma que caracterizan a dicha imagen y el nombre de la macro creada que es el nombre de un fichero XML. Esta información es guardada en un almacén que es un fichero en

[10] API (*Application Programming Interface*) Interface de Programación de Aplicaciones es el conjunto de funciones y procedimientos que ofrece una biblioteca para ser usada por otro software como una capa de abstracción.
[11] XML Extensible Markup Language o Lenguaje de Marcado Extensible.

formato CSV[12]. Por último la aplicación utiliza el fichero CSV para generar automáticamente recomendaciones para el retoque de las imágenes. La base para esta recomendación automática es la comparación de los valores de los histogramas de las imágenes.

4.2 *Arquitectura de la aplicación*

Para plasmar la arquitectura de la aplicación se ha elegido un diagrama de flujo de datos, en el que poder ver toda la aplicación completa. Para completar la información se muestra una especificación formal de cada proceso, y un diccionario de datos que aporta conocimiento más detallado.

4.2.1 Diagrama de módulos

Este diagrama da una visión general de la aplicación. Muestra los módulos operativos que la componen desde el punto de vista de un usuario, y la intercomunicación entre ellos.

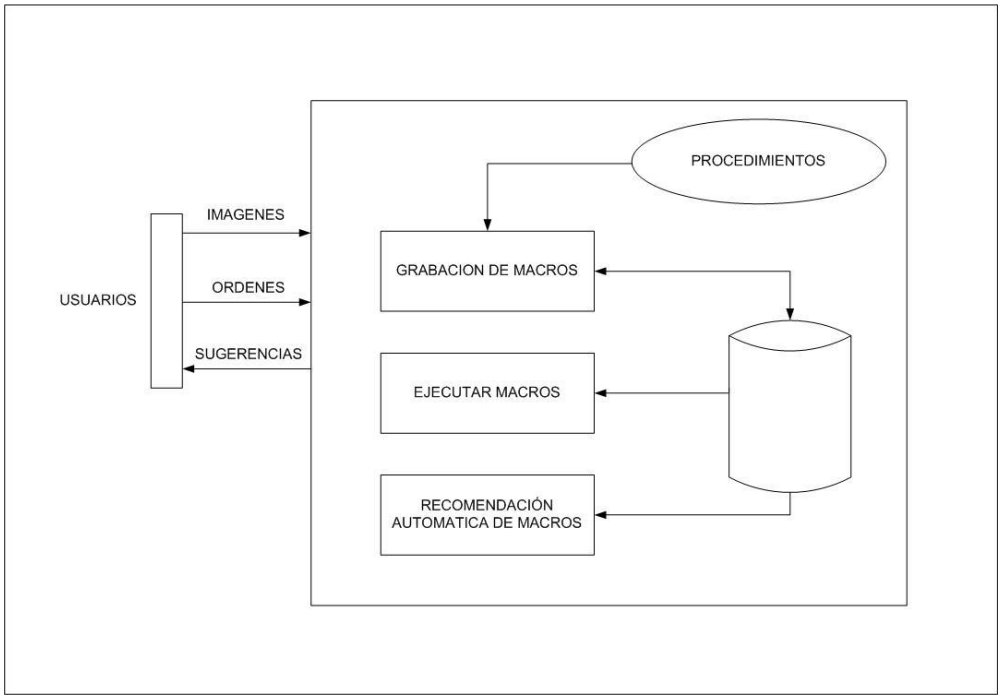

Figura 3. Diagrama de Módulos

[12] CSV Comma-Separate Values o Valores Separados por Comas.

4.2.2 *Diagrama de flujo de Datos*

La aplicación completa se compone de varios procesos (que son los *plug-ins*) que interactúan a través de ficheros. En el diagrama de flujo de datos de la Figura 4 se puede ver la aplicación completa desglosada en procesos. El proceso "`inicio`" y cualquiera de los procesos de "`procedimientos`" van introduciendo datos en los ficheros temporales. Cuando el usuario decide terminar una macro, el proceso "`fin`" pasa los datos definitivos a los ficheros permanentes. Los procesos "`macros`" y "`recomendaciones`" leen los datos de los ficheros principales para llevar a cabo su cometido. Para la explicación de su arquitectura se va a usar un diagrama de flujo de datos en el que se puede ver la aplicación completa desglosada en procesos.

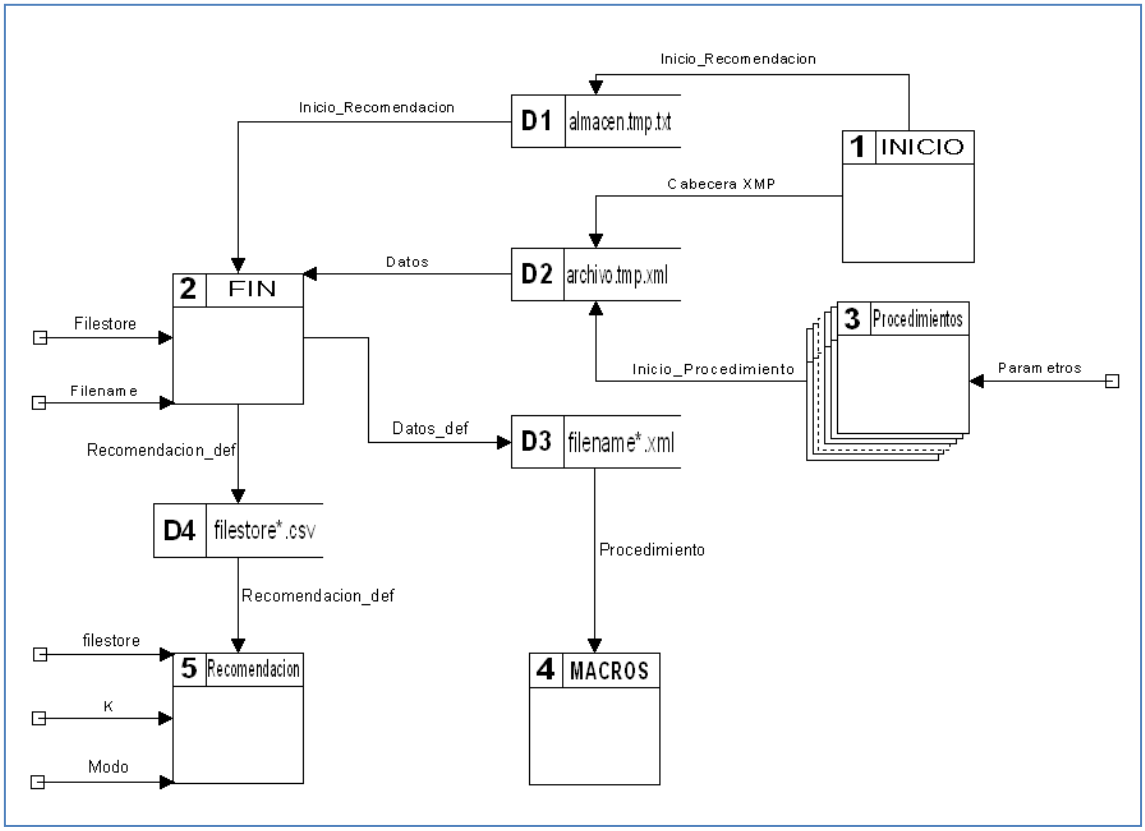

Figura 4. DFD Diagrama de flujo de datos.

4.2.3 Especificación de procesos del DFD

La especificación de los procesos aporta una visión rápida y elemental del funcionamiento de cada uno de ellos.

PROCESO 1. INICIO.

ENTRADA

Nombres de inicio de los ficheros temporales.

PROCESO

Crea e inicializa los ficheros temporales.

SALIDA

Cabecera XML, nombre de la imagen inicial y sus valores del histograma.

PROCESO 2. FIN.

ENTRADA

Todos los datos de los ficheros temporales. Y los nombres y ubicaciones de los ficheros definitivos.

PROCESO

Crea el fichero XML definitivo de la macro, lo cumplimenta y finaliza. Rellena el almacén definitivo de recomendaciones.

SALIDA

Datos que caracterizan la macro. Información de la recomendación que va a quedar almacenada.

PROCESO 3. PROCEDIMIENTOS.

ENTRADA

Parámetros específicos de cada procedimiento.

PROCESO

Aplica la funcionalidad a la imagen. Guarda los datos en el almacén temporal.

SALIDA

Nombre del procedimiento y valores de los parámetros.

PROCESO 4. MACROS

ENTRADA

Nombre del fichero a ejecutar y nombre del procedimiento a aplicar.

PROCESO

Aplica a la imagen las funcionalidades de la macro.

SALIDA

Crea fichero con la imagen modificada.

PROCESO 5. RECOMENDACIONES.

ENTRADA

Nombre y ubicación del fichero de recomendaciones. Valor de K y opción de modo. Datos de recomendaciones del almacén definitivo.

PROCESO

Devuelve la/s recomendación/es correspondientes.

SALIDA

La/s imagen/es con las recomendaciones aplicadas.

4.2.4 Diccionario de datos

El diccionario de datos se compone de una descripción detallada de los ficheros y de los flujos de datos que componen la aplicación. Consiste en especificar los datos contenidos en los ficheros que se emplean, tanto temporales como definitivos, y también en los flujos de datos que existen entre los procesos que se han expuesto en el Diagrama de Flujo de Datos.

FICHEROS

D1. almacen.tmp.txt = nombre_imagen + valores_histograma
D2. archivo.tmp.xml = cabecera_XML
D3. filename*.xml = cabecera_XML + procedimiento + parametros
D4. filestore*.csv = nombre_imagen + valores_histograma + nombre_fichero_XML

FLUJOS DE DATOS

- Inicio_Recomendacion = nombre_imagen + valores_histograma
- Cabecera_XML = cabecera_XML
- Inicio_Recomendacion = nombre_imagen + valores_histograma
- Datos = cabecera_XML + procedimiento + parámetros
- Filestore = nombre_fichero_CSV
- Filename = nombre_fichero_XML
- Datos_def = cabecera_XML + procedimiento + parámetros
- Inicio_Procedimiento = procedimiento + parámetros
- Parametros = parámetros
- Recomendación_def = nombre_imagen + valores_histograma + nombre_fichero_XML
- Procedimiento = procedimiento + parámetros
- K = valor_K
- Modo = opción_modo

Para no dejar lugar a dudas se explica a continuación el significado de todos los datos aparecidos:

- **"nombre_imagen"**: Se refiere al nombre de una imagen, normalmente el nombre de un fichero JPG.
- **"valores_histograma"**: Son 27 valores numéricos extraídos del histograma de cada imagen.
- **"cabecera_XML"**: Es la cabecera debe contener un fichero para que se identifique como un fichero XML `<?xml version="1.0" encoding="UTF-8" ?>`
- **"procedimiento"**: El nombre del procedimiento.
- **"parámetros"**: Valores usados por un procedimiento.
- **"nombre_fichero_XML"**: Nombre del fichero XML que contienen una macro.
- **"nombre_fichero_CSV"**: Nombre del fichero CSV de recomendaciones en uso.
- **"valor_K"**: Valor numérico que representa el numero de sugerencias solicitado por el usuario.
- **"opción_modo"**: Valor numérico que representa el modo elegido por el usuario.

4.3 *Descripción a alto nivel de cada modulo*

A continuación se explican detenidamente todo lo referente a los *plug-ins* de los que se compone la aplicación.

4.3.1 Estructura de un plug-in

En cada *plug-in* o complemento se pueden diferenciar tres partes, la primera es para importar los módulos que contienen funciones útiles para usar en el código. El modulo que siempre se tiene que importar

es: "`from gimpfu import*`". La siguiente parte diferenciada es la función que es el propósito del *plug-in*, es decir, lo que va a ejecutar cuando se pulsa el botón de aceptar. Y por último está una llamada a la función principal "`register`", que es la encargada de decirle a GIMP lo que tiene que hacer.

Lo primero que hace el *plug-in* es importar los módulos necesarios. Hay muchos módulos, unos son comunes para todas las plataformas y otros no. Por eso, lo mejor es consultar la ayuda que ofrece *Python*.

A continuación se da el nombre de la función en la que se desarrolla todo lo que se quiere que haga el *plug-in*, el cual lleva el mismo nombre. La función recibe como parámetros las variables que se van a necesitar y es del tipo: "`def Resize (img, drawable, width, height):`". Los dos primeros parámetros que se pasan a esta función son obligatorios siempre, y son "`img`" que es la imagen, y "`drawable`" que es la capa activa. El resto de parámetros pueden ser de cualquier tipo, y se obtienen mediante el interface de usuario de la aplicación. Hay otra variable más llamada "pdb", que es una variable global que no hay que pasarla como parámetro. Esta variable es la que da acceso a todas las funciones de GIMP a través del API. Esto es lo que da la posibilidad de usar las funcionalidades que ya existen de GIMP, pero por medio de código. Las funciones de GIMP disponibles para usar con la variable "pdb", están en el propio GIMP. Se puede encontrar la explicación de estos procedimientos y los parámetros que necesitan en el menú de GIMP "`ayuda/visor de procedimientos`".

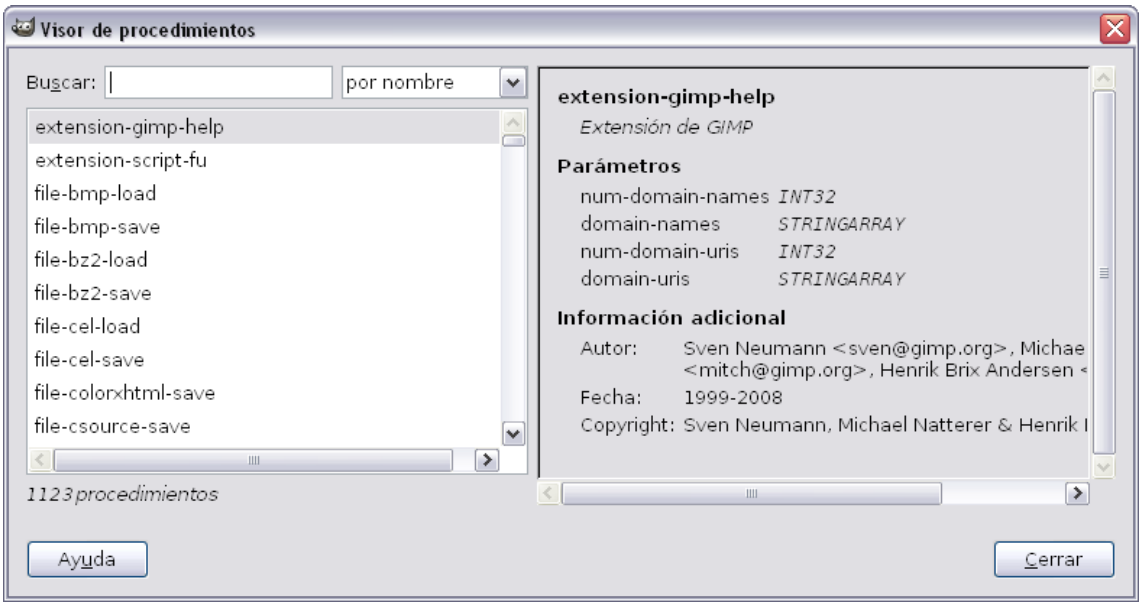

Figura 5. Visor de procedimientos de GIMP

La última parte es la llamada a la función principal "`register`" donde se pone toda la información necesaria para que GIMP pueda ejecutar el *plug-in*. El primer parámetro es el nombre del *plug-in*, y debe ser un nombre único, que no se repita en ningún otro *plug-in* y que no tenga ni acentos ni espacios ni caracteres especiales. En las siguientes líneas van parámetros descriptivos de lo que va a hacer el *plug-in*. Lo que se pone en la primera línea aparecerá en el cuadro de dialogo como descripción de la función, y en la segunda línea se pone la descripción que se quiere que aparezca en la ayuda contextual de GIMP. A continuación aparecen el nombre del autor y el año de la creación. Después está la ruta en el menú de GIMP. Este parámetro es importante, ya que determina el lugar del menú de GIMP desde donde se va a ejecutar la función creada. Se entiende como una ruta de directorios en la que cada "/" es un submenú en el menú de GIMP. La ruta se puede empezar de dos formas: con `<Image>` de manera que la nueva funcionalidad se sitúa en el menú de la imagen abierta, o con `<Toolbox>` siendo esta vez el menú elegido el de la caja de herramientas de GIMP.

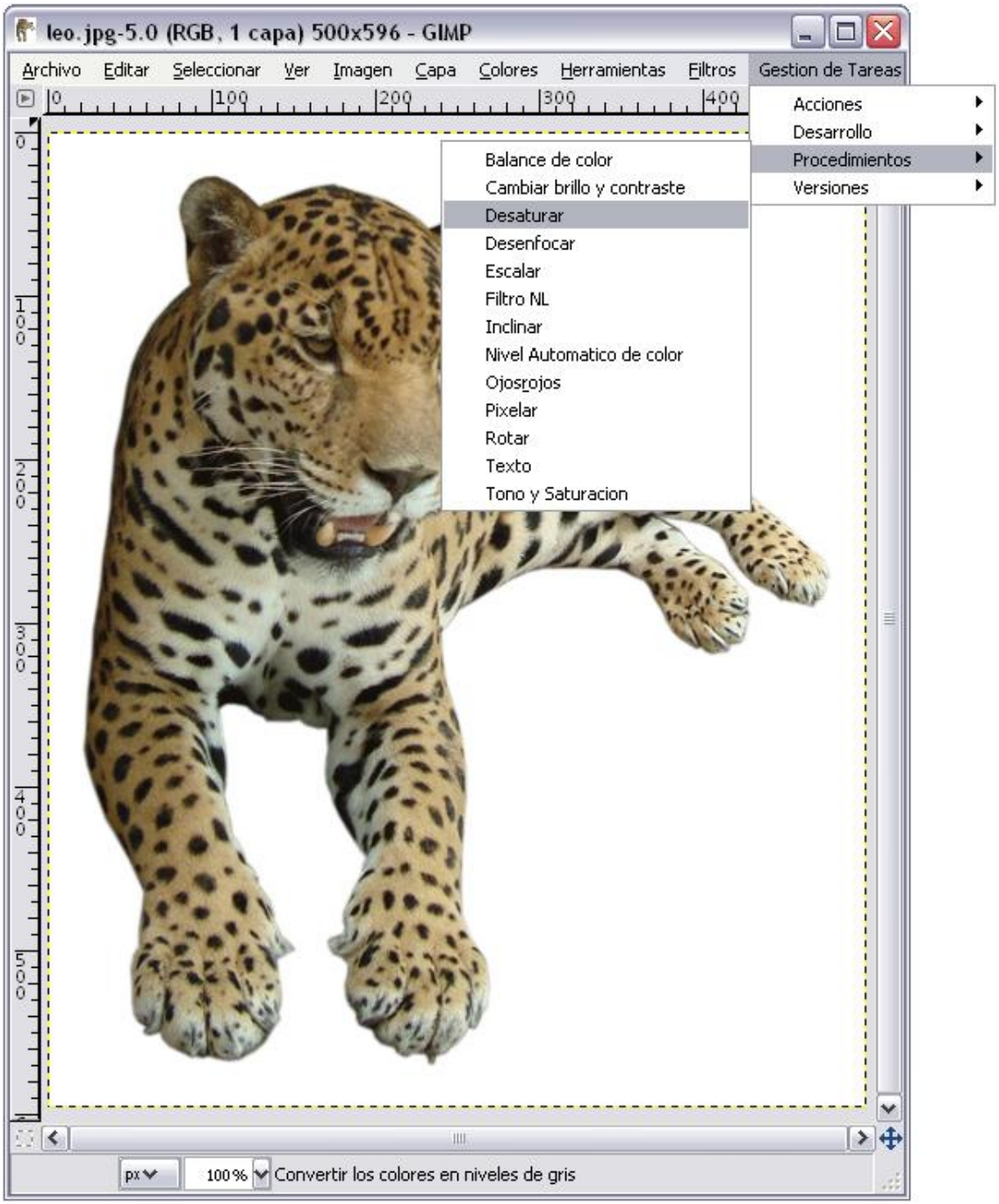

Figura 6. Menú `<Image>` de GIMP.

En la Figura 6 se ve que para ejecutar la función `"desaturar"` hay que ir al menú de la imagen abierta en GIMP y seguir por los submenús indicados. Y a continuación la línea de código necesaria.

```
"<Image>/Gestion de tareas/Procedimientos/Desaturar"
```

El siguiente parámetro que aparece indica el tipo de imagen. Y a continuación se ponen los parámetros del *plug-in*, que son los componentes que se van a usar en la ejecución; es decir, son las cajas de texto o las barras de deslizamiento que van a aparecer en el cuadro de diálogo que se abre al ejecutar el *plug-in*, para obtener la información del usuario. Existen varias opciones, y son instrucciones de este tipo:

```
(PF_SPINNER, "width", "Ancho", 200, (0, 1000, 1))
```

Figura 7. Cuadro de dialogo.

En este caso "`width`" es la variable que se pasa como parámetro a la definición de nuestra la función del *plug-in*. "Ancho" es el nombre que aparecerá en la ventana, y a continuación los valores de la variable, que son 200 por defecto, y puede tomar el valor de 0 a 1000 en intervalos de 1. Se pueden poner todos los parámetros que se necesiten, y hay varios tipos, para recoger valores numéricos, o seleccionar entre varias opciones, o un nombre de fichero o todo el *path*, etc.… Luego tenemos un parámetro de resultados que se deja en blanco, y por último se hace una llamada al nombre de la función objeto del *plug-in*.

Usando esta estructura básica es como se puede hacer crecer la aplicación a base de complementos.

4.3.2 Procedimientos

Estos *plug-ins* ejecutan funcionalidades sobre las imágenes. Todos están basados en el mismo tipo de código, que consiste en la llamada a un procedimiento de GIMP mediante la función especial "`pdb`". Se encuentran en el menú "`Gestion de Tareas/Procedimientos/...`". La ejecución del código se inicia con la llamada a la función "`pdb`" con el nombre del procedimiento correspondiente y junto con los parámetros obtenidos del usuario mediante la interface gráfica.

El *plug-in* continúa con un tratamiento de excepción para abrir el fichero temporal donde se va a almacenar la macro. En este fichero se guardan de forma adecuada el nombre del procedimiento y los valores de los parámetros usados. Los valores de los parámetros hay que guardarlos de manera que luego los podamos recuperar y usar para ejecutar la funcionalidad correspondiente; es decir, teniendo en cuenta los tipos usados. Por ejemplo, si una funcionalidad necesita como parámetros unas variables del tipo *integer*, hay que guardarlas como *integer*, y lo que es más importante, luego recuperarlas como *integer* también.

Una vez finalizada la ejecución, el resultado es la aplicación sobre la imagen del procedimiento elegido, y además, si se siguieron los pasos previos para crear una macro, en el fichero XML de la misma se habrán guardado el nombre del procedimiento y los valores de los parámetros.

A continuación se relacionan los procedimientos implementados y los nombres de los ficheros correspondientes:

"Balance de color" → color.py

"Cambiar brillo y contraste" → brillo.py

"Desaturar" → desaturar.py

"Desenfocar" → desenfocar.py

"Escalar" → escalar.py

"Filtro NL" → filtro_NL.py

"Inclinar" → inclinar.py

"Nivel Automatico de color" → nivel_auto.py

"Ojos rojos" → ojos_rojos.py

"Pixelar" → pixelizar.py

"Rotar" → rotar.py

"Texto" → add_texto.py

"Tono y Saturacion" → tono&saturación.py

4.3.3 INICIO.py

Este *plug-in* es el que se usa para iniciar la grabación de una macro. Cuando un usuario quiere crear una grabación de una o varias funcionalidades, ha de empezar por aquí, y se ejecuta desde el menú "Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro". El *plug-in* lo que hace es crear dos ficheros temporales en los que se va a almacenar todo lo necesario para la creación de la macro y para el almacén de recomendaciones. Estos

ficheros almacenan la información temporalmente, hasta que el resto de procesos, terminen de añadir los datos necesarios, y se pasen todos ellos a los ficheros definitivos.

El primer fichero temporal creado se denomina "`archivo.tmp.xml`" y es el inicio de lo que finalmente terminará siendo la macro. Es un fichero con formato XML, así que se crea el fichero y se introduce la cabecera XML para respetar el formato.

El otro fichero temporal creado se llama "`almacen.tmp.txt`". En este fichero se almacena la información relativa a la imagen original que va a ser retocada. Esta información es la que se pasará después al fichero almacén de recomendaciones que tiene formato CSV, así que es preparada para cumplir las reglas de dicho formato. Los datos que se guardan son el nombre del fichero de la imagen con la que se va a trabajar, que debe ir entre comillas, como por ejemplo "`imagen.jpg`". A continuación se guardan una serie de valores del histograma de dicha imagen obtenidos con el procedimiento de GIMP "`pdb.gimp_histogram(drawable,0,0,255)`". Estos valores son la media aritmética, la desviación estándar y la mediana, que son los tres primeros valores que devuelve el procedimiento. Después se almacenan el porcentaje de píxeles de cada rango. Los rangos que se han elegido son 8 rangos de 32 niveles para cada color rojo, verde y azul, lo que deja un total de 27 valores numéricos por cada imagen.

Estos valores se consiguen modificando los 3 argumentos numéricos del procedimiento "`pdb.gimp_histogram(drawable,0,0,255)`". El primer argumento se refiere al color, siendo 1=Rojo, 2=Verde y 3=Azul. Los otros 2 argumentos numéricos son el inicio y el final del rango del que se quiere conocer el porcentaje de píxeles. Se implementa con un bucle "`for`" doble, el primero para los 3 colores, y dentro de éste, el segundo para los 8 rangos de 32 niveles cada uno, tomando los valores de los argumentos para el procedimiento del

histograma los valores de inicio y fin de rango adecuados tal y como se pueden observar en la Figura 8.

	8 rangos de 32 niveles	
	nivel - start	nivel - end
rango 1	0	31
rango 2	32	63
rango 3	64	95
rango 4	96	127
rango 5	128	159
rango 6	160	191
rango 7	192	223
rango 8	224	255

Figura 8. Rangos de los niveles de luminosidad.

A continuación se muestra un ejemplo real de una imagen, seguida de su histograma en GIMP y al final los valores del histograma que recupera este *plug-in*.

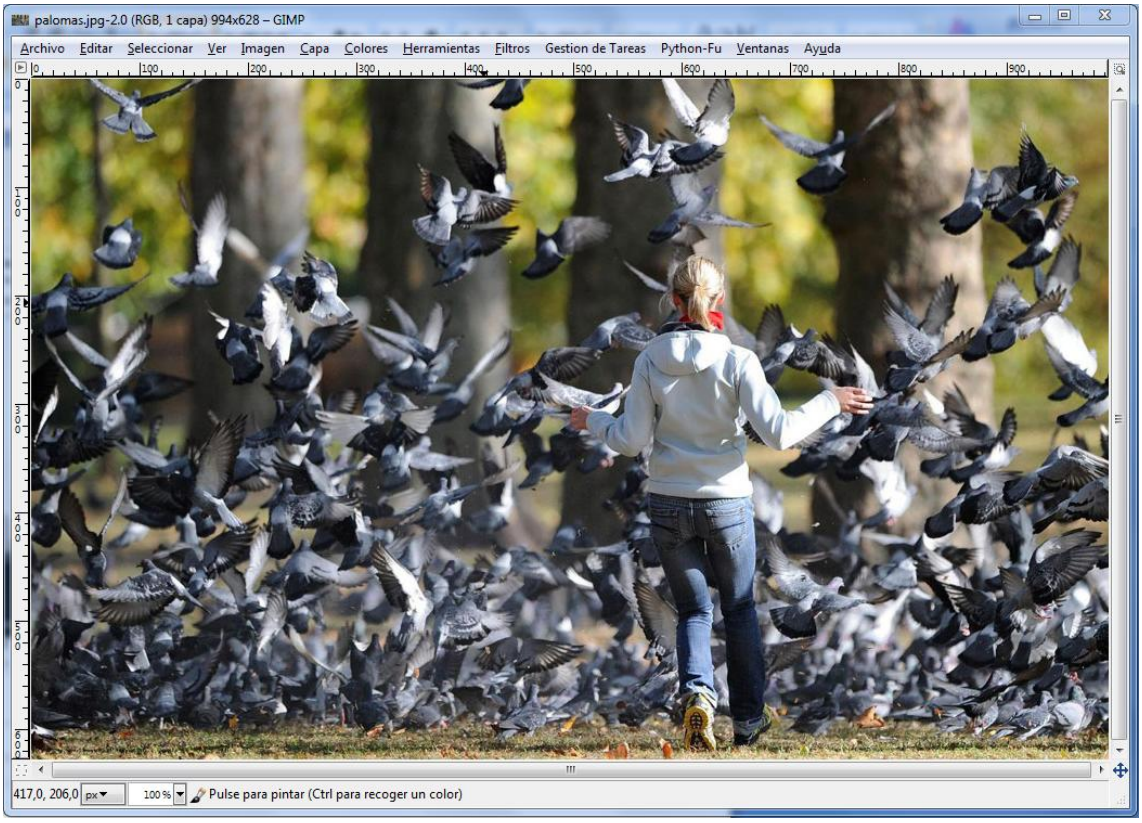

Figura 9. Imagen "`palomas.jpg`" editada con GIMP.

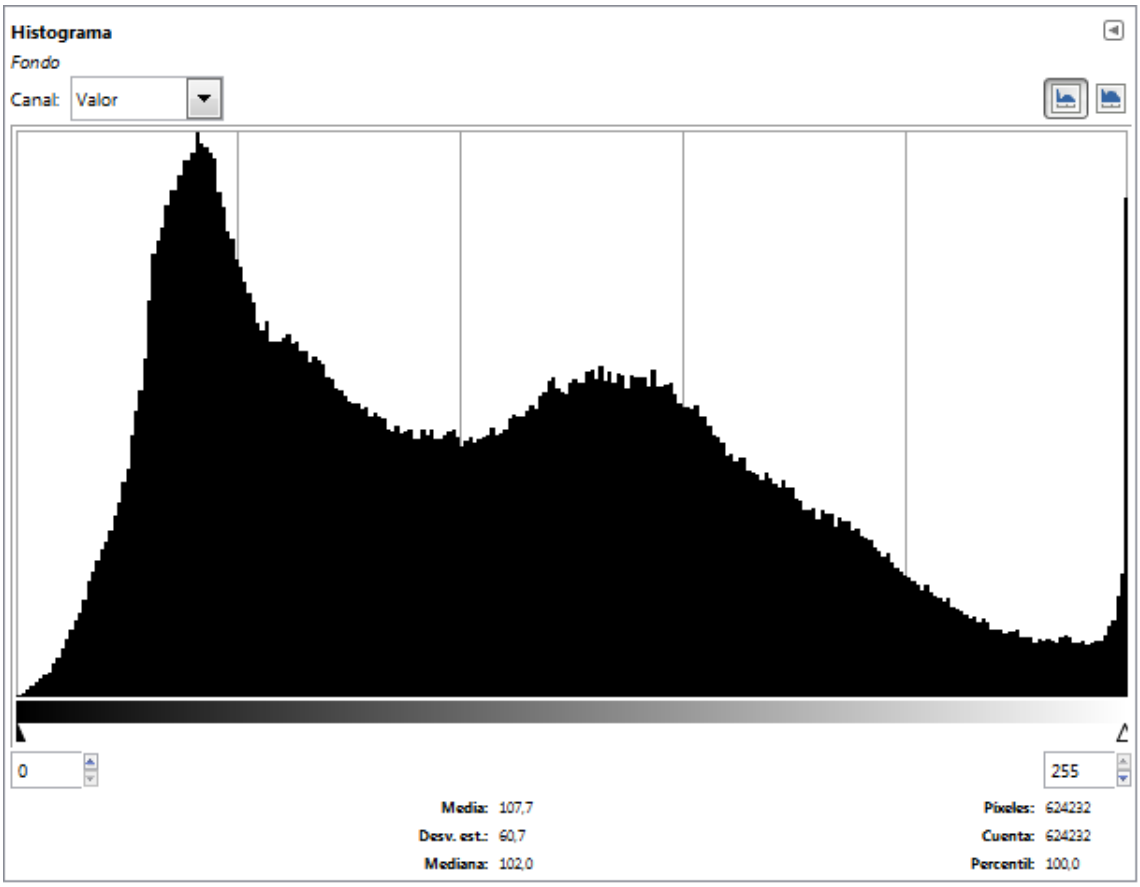

Figura 10. Histograma en GIMP de la imagen "palomas.jpg"

Datos recuperados por el *plug-in* de la imagen "palomas.jpg":

"palomas.jpg";107.667;60.672;102;112;247;149;153;155;100;50;31;103;
252;154;157;162;100;41;27;133;299;188;138;109;66;31;33;

4.3.4 FIN.py

Una vez se ha terminado de ejecutar los procedimientos para manipular la imagen, hay que finalizar la grabación de la macro, y este es el objetivo de este *plug-in*. Se ejecuta desde el menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro". Al usuario se le pide el nombre y ubicación del fichero XML que quiere que contenga la nueva macro, y del fichero CSV donde quiere que se almacenen los valores de la imagen manipulada y el nombre de la macro.

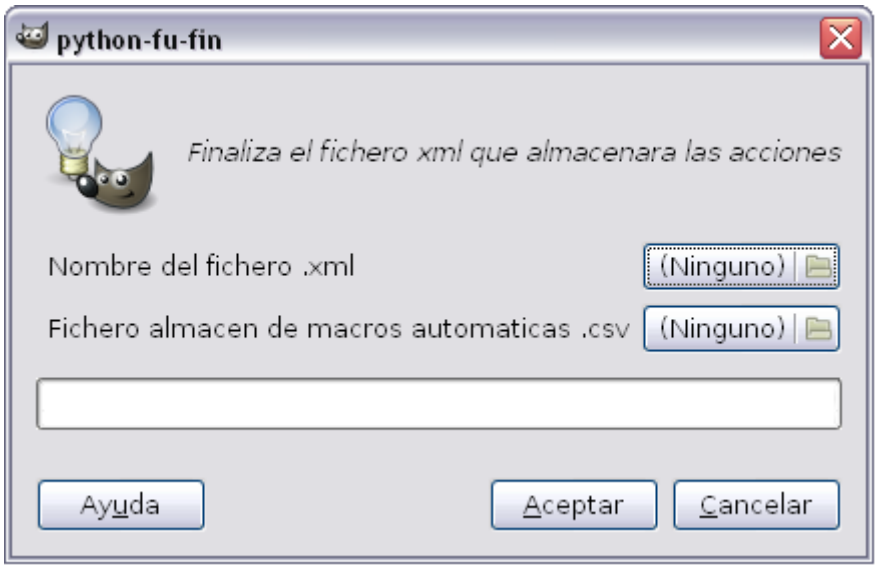

Figura 11. Cuadro de dialogo del *plug-in* "`fin.py`"

Este *plug-in* abre el fichero temporal "`archivo.tmp.xml`" en el que se han ido almacenando los datos de las acciones de los procedimientos, y transfiere línea por línea toda la información al fichero XML definitivo indicado por el usuario. Una vez terminado, se cierra el fichero XML definitivo y el fichero temporal "`archivo.tmp.xml`" se cierra y se elimina.

Después se abre y lee el fichero temporal "`almacen.tmp.txt`" en el que se han almacenado los valores de la imagen manipulada, y se escriben estos datos en el fichero CSV que introduce el usuario. También se añade como último dato en el fichero CSV el nombre del fichero XML definitivo que se acaba de crear. Generalmente un usuario usa un único fichero CSV de recomendaciones. Cada vez que se finaliza el proceso de grabación de una macro con este *plug-in*, el fichero CSV se incrementa con una nueva recomendación. Una vez terminado, se cierran ambos ficheros, y además el fichero temporal "`almacen.tmp.txt`" se elimina.

4.3.5 MACROS.py

Con este *plug-in* el usuario puede ejecutar las macros anteriormente creadas, es decir, los ficheros XML. Se encuentra en el menú "`Gestion de tareas/Acciones/Ejecutar macros`". Cuando se ejecuta, a través del cuadro de dialogo, el usuario introduce el nombre del fichero XML que quiere ejecutar.

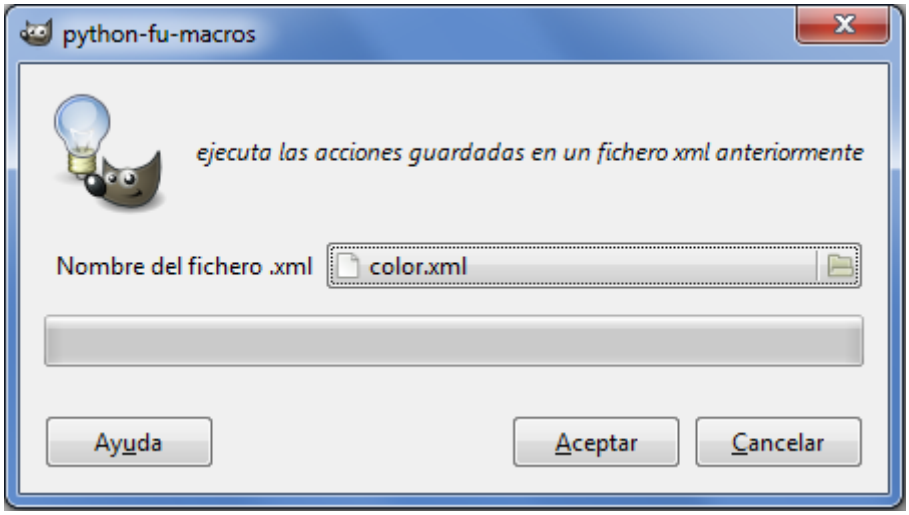

Figura 12. Cuadro de dialogo del *plug-in* "`macros.py`"

Lo primero que hace el *plug-in* es duplicar la imagen con la que el usuario está trabajando, y se la da el nombre del fichero XML seleccionado por el usuario. Sobre esta copia se van a aplicar los cambios.

Luego se abre y se lee el fichero XML línea por línea. Después un bucle recorre estas líneas creando una lista llamada "`procedimientos`", que se compone de tantas listas como acciones diferentes tenga la macro, y cada una de estas listas está formada por el nombre del procedimiento de la acción seguido de los valores de los parámetros aplicados.

A continuación se muestra una lista de ejemplo, que contiene entre corchetes una primera acción solo formada por el procedimiento `['gimp_levels_stretch']` sin parámetros, y otra segunda acción

entre corchetes, formada por el nombre del procedimiento `'gimp_color_balance'` seguido por los valores de los parámetros:

```
[['gimp_levels_stretch'],['gimp_color_balance','2','False',
'-30.0','10.0','0.0']]
```

Por último, se recorre la lista y se forman las instrucciones de ejecución usando la función `"pdb"` junto con el nombre del procedimiento y los valores de los parámetros.

4.3.6 RECOMENDACIÓN.py

Este *plug-in* permite al usuario solicitar al sistema una recomendación de la o las acciones a aplicar a una nueva imagen a partir de las acciones aplicadas a las imágenes previas. Se accede a través del menú `"Gestion de tareas/Acciones/Recomendacion Automatica de macros"`.

Desde el cuadro de diálogo correspondiente, el usuario introduce el nombre y dirección del fichero CSV que desea usar. Este es el fichero almacén que contiene todos los nombres de las macros, asociados a los datos de las imágenes con las que se han grabado. También se introduce el valor de K, que es el número de recomendaciones que solicita el usuario. Y por último se selecciona el modo, que es la forma de extraer las sugerencias solicitadas, como se explica detalladamente un poco más adelante (Figura 13).

Figura 13. Cuadro de dialogo del *plug-in* "`recomendacion.py`"

Este *plug-in* compara los valores del histograma de la imagen que se quiere manipular, con los valores de los histogramas de todas las imágenes asociadas con los nombres de las macros que están almacenadas en el fichero CSV almacén de recomendaciones. Toda esta comparativa va a dar como resultado una lista de valores dados por el cálculo de la distancia euclidiana, donde cada valor se asocia con el nombre del fichero XML correspondiente. Y en función del valor de K y del modo seleccionado, se eligen unos valores de la lista, y se proponen como recomendación las macros asociadas a dichos valores.

La comparativa se hace usando la distancia euclidiana entre dos puntos de n coordenadas cada uno. Estos puntos son las imágenes A y B, definidas por los 27 valores numéricos obtenidos de sus respectivos histogramas. Así el cálculo de la distancia euclidiana entre las dos imágenes es:

$$\sqrt{\sum_{i=1}^{n}(ImgA_i - ImgB_i)^2}$$

La distancia euclidiana es una forma sencilla de calcular la distancia entre dos puntos, pero con el inconveniente de que la distancia resultante es sensible a las unidades de medida de las variables, es decir, las diferencias medidas con valores altos contribuirán en mucha mayor medida que las diferencias entre variables con valores bajos. Esto se soluciona con la normalización que es un método que, en general, mejora los resultados.

Un ejemplo de la normalización a partir de la formula de las imágenes

$$\sqrt{\sum_{i=1}^{n}(ImgA_i - ImgB_i)^2}$$

Si se considera

$$ImgA_i = \frac{ImgA_i - min}{Max - min} \qquad ImgB_i = \frac{ImgB_i - min}{Max - min}$$

Los valores Max y min son el máximo y mínimo que pueden alcanzar las variables y el mínimo posible es 0 $(min = 0)$, entonces

$$\sqrt{\sum_{i=1}^{n}\left(\frac{ImgA_i}{Max} - \frac{ImgB_i}{Max}\right)^2} = \sqrt{\sum_{i=1}^{n}\frac{1}{Max^2} * (ImgA_i - ImgB_i)^2}$$

Por lo tanto la formula general que usa este *plug-in* para el cálculo de la distancia euclidiana entre una imagen nueva (actual) y cada una de las imágenes previas (datos) es:

$$\sum_{i=0}^{n}\frac{1}{M_i^2} \times (datos_i - actual_i)^2$$

"M" es el coeficiente usado para hacer la normalización. No se usa el mismo valor de "M" para todos los sumandos. Para la media, la desviación estándar y la mediana el valor de "M" es 256, y para el resto "M" es igual a 100.

Cada una de las variables de la formula anterior se implementa con una lista de n=27 valores cada una. La lista de los coeficientes "M", la lista de "datos", que contiene los valores del histograma de las imágenes previas del fichero CSV de recomendaciones y la lista "actual", que contiene los valores del histograma de la imagen nueva, sobre la que el usuario solicita la recomendación.

Con estas listas se hace el cálculo de la distancia euclidiana entre la imagen de cada recomendación y la imagen actual, de manera que se obtiene una lista "`resultados_comparaciones`", en la que cada elemento se compone del valor numérico resultante de cada operación y el nombre del fichero XML que representa la macro correspondiente a cada imagen.

A continuación se ordena la lista usando el método de la burbuja. Se ha elegido éste porque se necesita un método de ordenación interna, ya que se trata de ordenar un elemento de almacenamiento que está siendo usado en fase de ejecución en memoria interna. Hay varios métodos de ordenación interna, clasificados en directos y avanzados, y el método de la burbuja es un método de ordenación directa, que es rápido de implementar, fácil y eficaz. Su desventaja viene cuando se usa con listas muy grandes. Pero, como este proyecto está en fase experimental y no usa grandes cantidades de datos, se considera que es un método adecuado. En el caso de ampliar el proyecto, el método de ordenación debería ser objeto de estudio para hacer la elección del mejor método de ordenación, valorando entre otros aspectos la cantidad de datos manejados, y la influencia en la velocidad de proceso.

La elección de los nombres de los ficheros XML de la lista que se van a usar como recomendaciones se obtienen con la parte del código que está representada en el diagrama de flujo de la Figura 14.

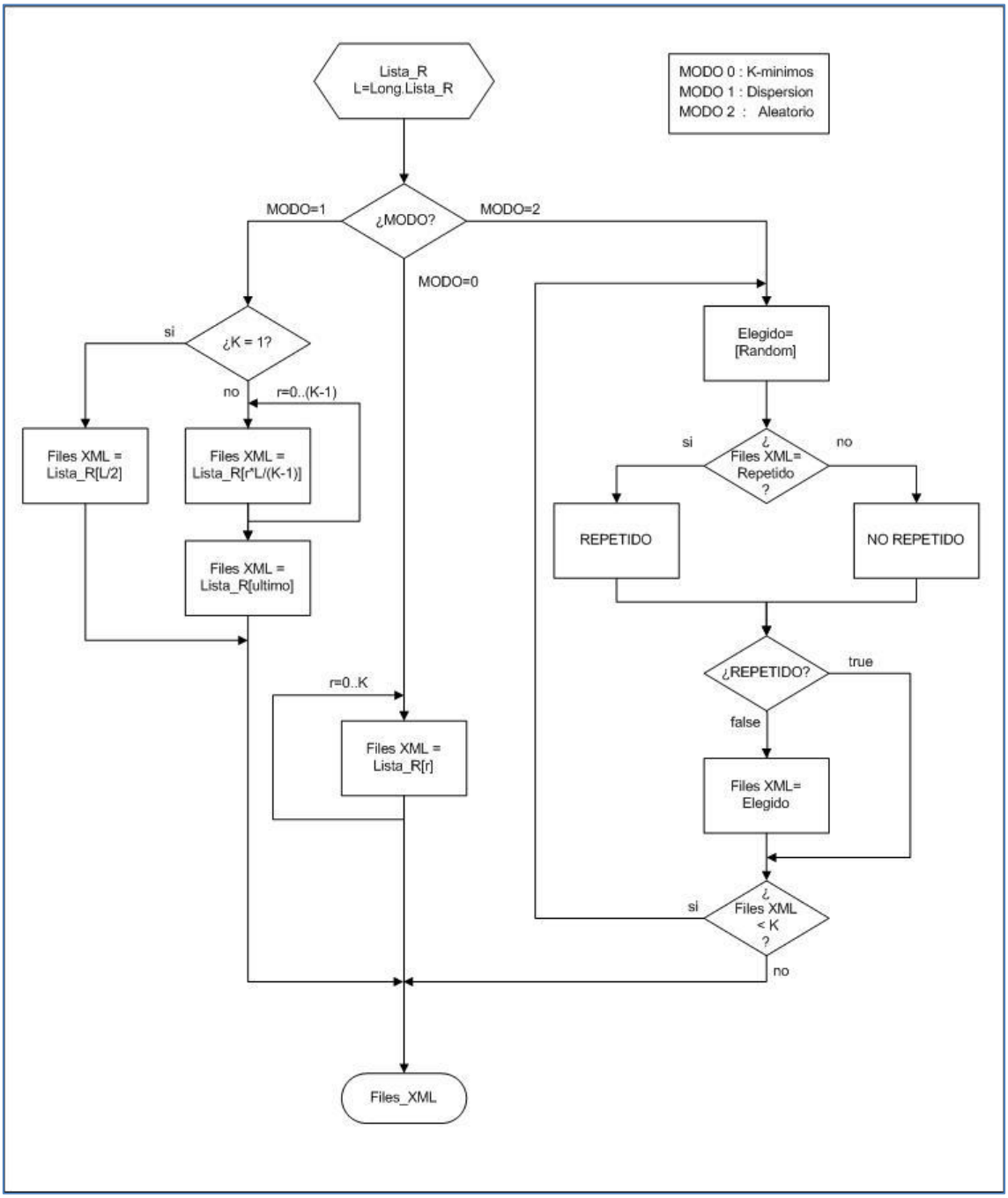

Figura 14. Diagrama de flujo de la selección de modo.

Esta elección depende del valor de K y del modo seleccionados por el usuario. Así se distingue entre los modos 0=K-mínimos, 1=dispersión

y 2=aleatorio. Y para cada uno de estos modos se usara el valor de K, que indica la cantidad de recomendaciones que solicita el usuario.

El modo K-mínimos consiste en devolver al usuario K macros para que las aplique a su imagen. El criterio para elegir las macros es que sus imágenes asociadas sean las más parecidas a la imagen actual o lo que es lo mismo, las que tengan la menor distancia euclidiana. Se implementa con un bucle en el que se toman los K primeros elementos de la lista "`resultados_comparaciones`" que está ordenada de menor a mayor distancia euclidiana, y se incluyen en la lista definitiva de nombres de ficheros XML "`lista_xml`".

El modo de dispersión es algo más complejo. Este modo ofrece las K recomendaciones más dispersas en todo el contenido de la lista de recomendaciones. Así si se solicita K=1 recomendaciones, el programa devolverá la recomendación que ocupe el centro de la lista. Si K=3 el programa propondrá la recomendación primera, la central y la última, y así sucesivamente. Para implementarlo, se distingue entre un valor de K=1 y el resto de valores. Si K = 1 entonces se devuelve la recomendación central. Y para el resto de valores de K (no se contempla como valores válidos de K ni el 0 ni números negativos, ya que se impiden estas opciones mediante la recogida de parámetros: `(PF_SPINNER, "K", "Valor de K", 1, (1, 1000, 1))`) se usa un bucle que devuelve las recomendaciones que estén en las posiciones de 0 a K-1 en intervalos de ([`numero_de_recomendaciones`/K-1]). Con esto se obtienen todas las recomendaciones, menos la última, la cual se asigna directamente usando el último elemento.

Por último el modo aleatorio devuelve K recomendaciones distintas, elegidas aleatoriamente. La implementación consiste en obtener mediante la función "`random`" tantos números aleatorios como indica K, y comprobar que no se repiten estos valores. Los números

resultantes indican la posición del elemento a elegir de la lista "`resultados_comparaciones`", y con los nombres del fichero XML se crea la lista "`lista_xml`".

Ahora se ejecuta a través de la función "`pdb`" el *plug-in* macros.py las veces que indica el valor de K, con los distintas macros referidas por los ficheros XML de la lista "`lista_xml`". El resultado es que se generan K-imágenes nuevas que son modificaciones de la imagen actual. El usuario puede elegir la o las que quiera de ellas guardándolas como nuevos ficheros de imagen.

4.4 *Manual de usuario*

Para que la aplicación funcione lo primero que se debe hacer es instalar GIMP y el resto de elementos necesarios tal y como se indica en el apartado 10.1 de este mismo documento.

La aplicación creada en este proyecto está totalmente integrada en GIMP, cuya interface gráfica tiene un sistema de ejecución en varias ventanas. La ventana principal o la de la imagen es similar a la de cualquier otro programa conocido, con los submenús típicos de archivo, ver, edición etc... y otros submenús más específicos. En este menú principal se encuentra el submenú para el uso de esta aplicación, llamado "`Gestion de Tareas`", y sobre el que se va a centrar el manual, remitiendo al usuario para el uso general de GIMP a los manuales del propio programa que se pueden encontrar en la página oficial http://www.gimp.org.

Para empezar a trabajar con esta aplicación es necesario abrir alguna imagen con GIMP. Para ello se accede al menú "`Archivo/abrir`" y se elige una imagen.

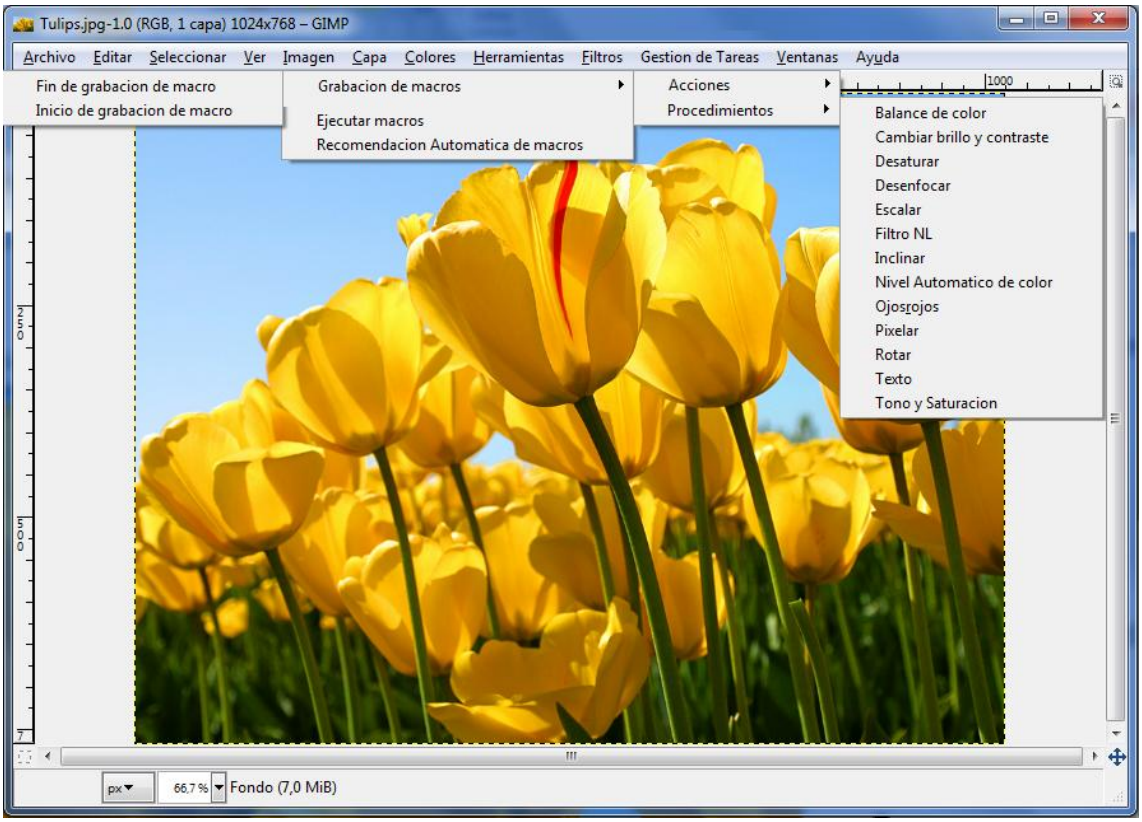

Figura 15. Menú de la aplicación.

En la Figura 15 se muestra la estructura completa del menú de la aplicación.

El menú principal "`Gestion de Tareas`" ofrece dos opciones principales: "`Acciones`" se refiere a todo lo relacionado con las macros, y "`Procedimientos`" se refiere a los procedimientos que se pueden aplicar a las imágenes.

A continuación se explican las operaciones que permite realizar la aplicación.

4.4.1 Uso de Procedimientos

En "Gestion de Tareas/Procedimientos/..." aparecen los procedimientos que se pueden usar con la imagen. Cada uno de ellos aplica un efecto distinto sobre la imagen, y también mediante una interface gráfica diferente, permiten modificar los parámetros específicos. Para ejecutarlos simplemente se acciona el procedimiento, y se modifican los parámetros.

Seguidamente se describen todos ellos.

-"Balance de color".- Permite modificar el balance de color de la imagen. Se puede modificar el rango en el que actúa, si conserva la luminosidad de la imagen, el balance de color entre cian/rojo, magenta/verde y amarillo/azul.

-"Cambiar brillo y contraste".- Para cambiar el brillo y el contraste de la imagen mediante una barra deslizante para cada uno.

-"Desaturar".- Se usa para quitar los colores de una imagen, usando una de las tres fórmulas que se ofrecen.

A continuación se muestra un ejemplo del funcionamiento de este procedimiento.

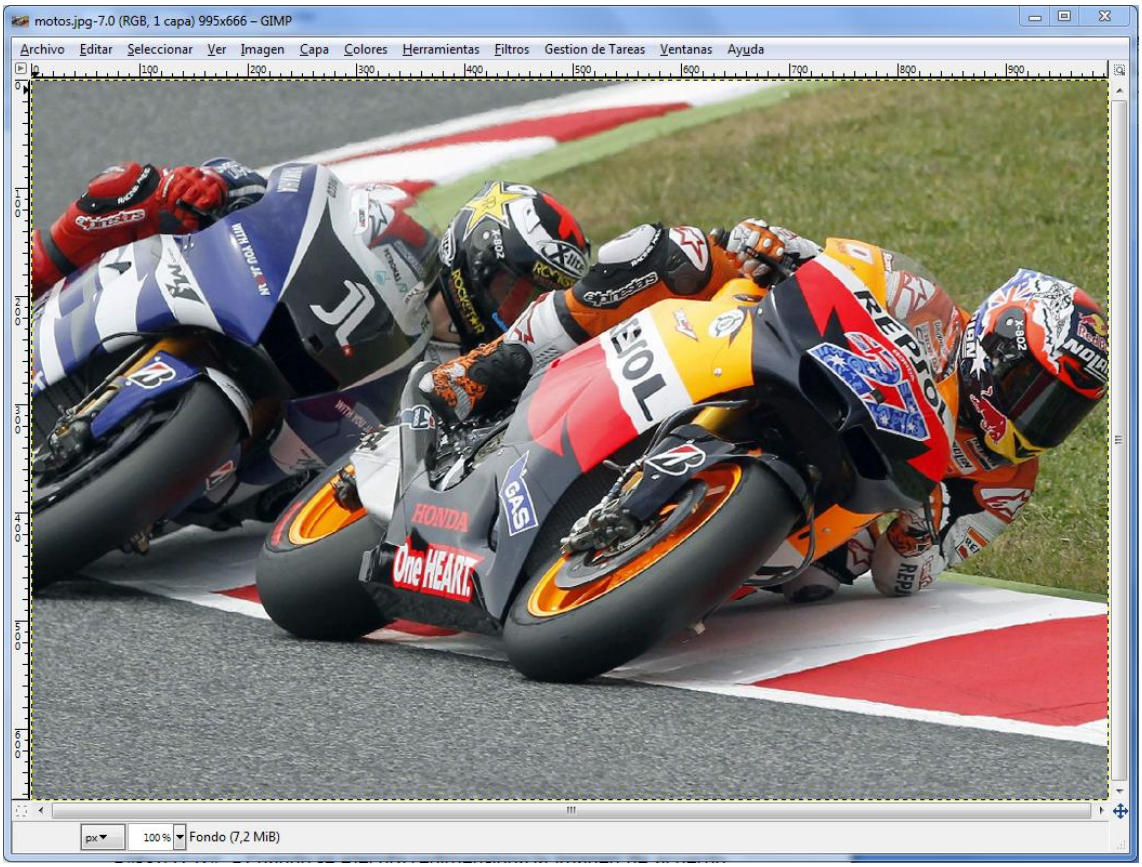

Figura 16. Imagen "motos.jpg" antes de aplicar "Desaturar"

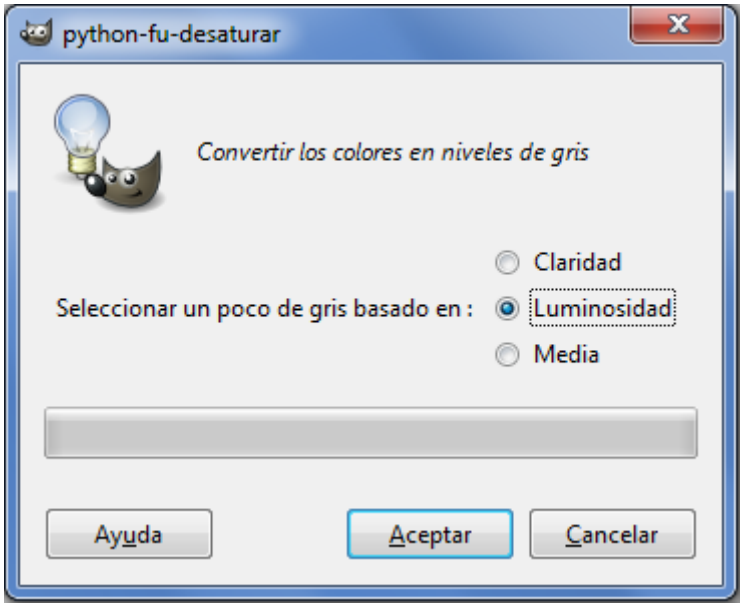

Figura 17. Cuadro de Diálogo del procedimiento "Desaturar".

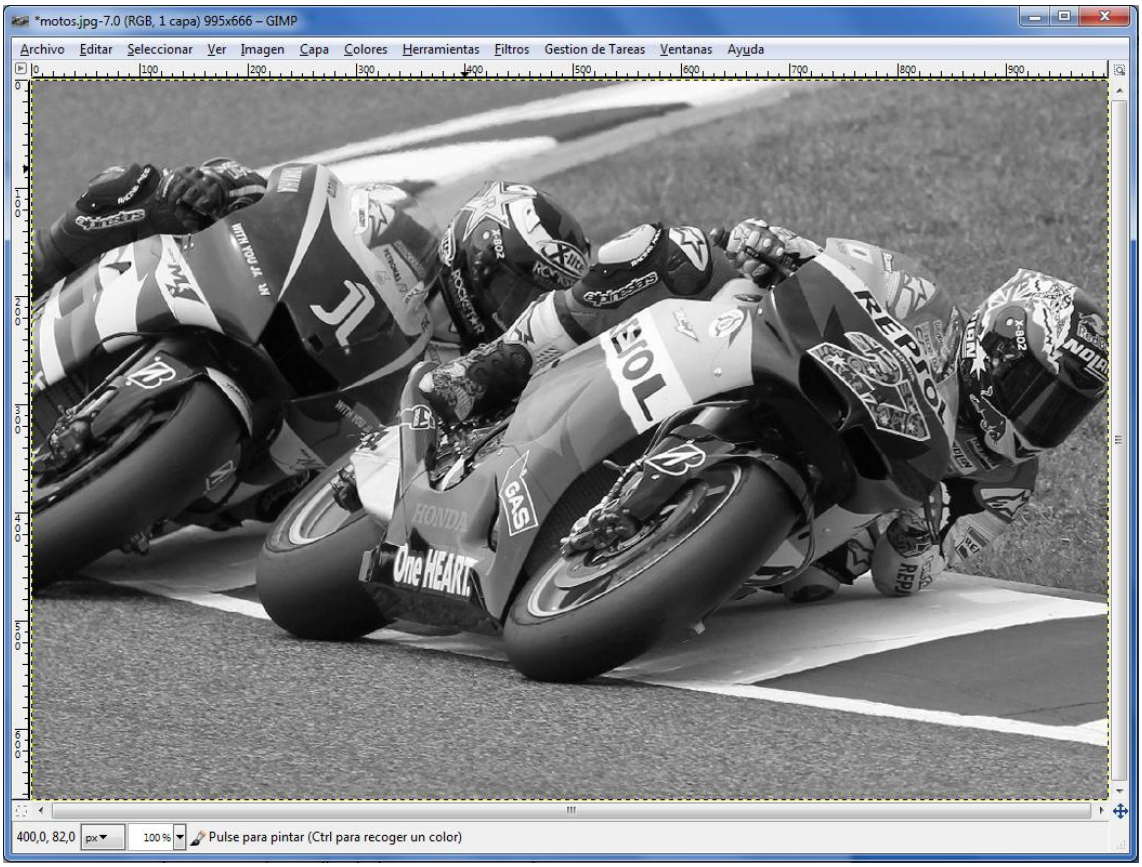

Figura 18. Imagen "motos.jpg" después de aplicar "desaturar".

-"Desenfocar".- Desenfoca la imagen; no admite parámetros.

-"Escalar".- Cuando se ejecuta redimensiona la imagen de acuerdo con el nuevo ancho y alto dados como parámetro.

-"Filtro NL".- Aplica a la imagen el filtro NL con varios parámetros modificables. NL viene de "No Lineal". Este filtro une las funciones de realce de suavizado, desparasitado y enfoque. Funciona sobre toda la imagen, no sobre selecciones. (Gimp).

A continuación se muestra un ejemplo del funcionamiento de este procedimiento.

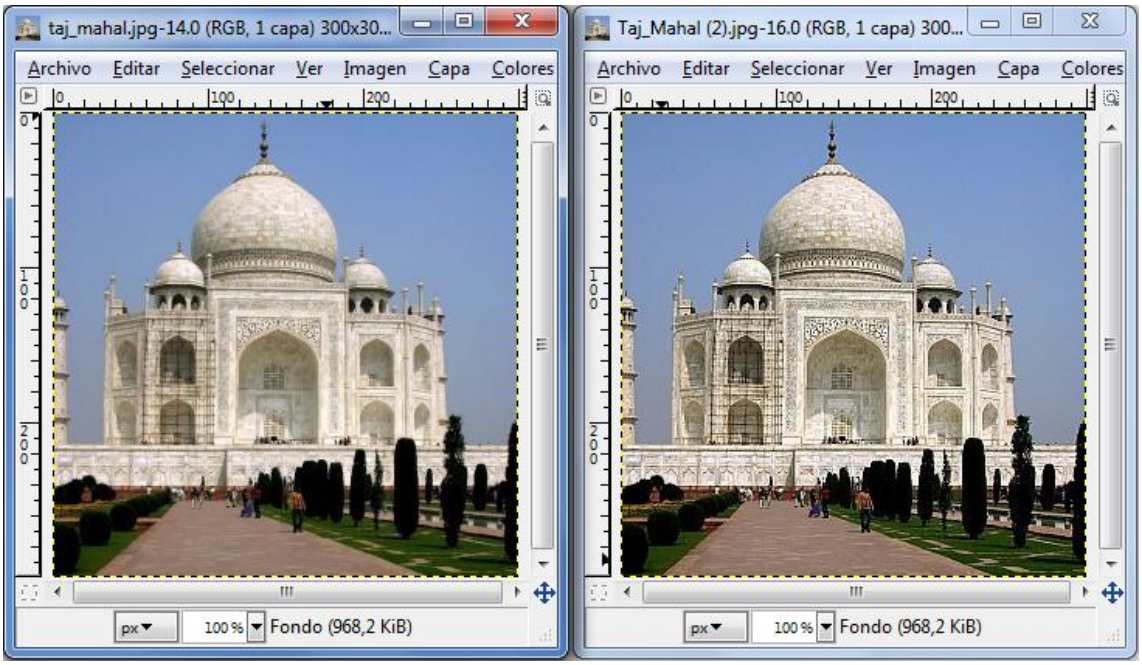

Figura 19. Antes y después de aplicar "filtro NL".

-"Inclinar".- Con este procedimiento se inclina la imagen. Se puede elegir la orientación de la inclinación, la magnitud de la misma y cómo recortar los resultados obtenidos.

-"Nivel Automatico de color".- Modifica automáticamente los niveles de intensidad de los colores de la imagen.

-"Ojos rojos".- Sirve para quitar el efecto de ojos rojos causado por los flashes de las cámaras.

-"Pixelar".- Permite modificar la altura y anchura de los píxeles de una imagen.

A continuación se muestra un ejemplo del funcionamiento de este procedimiento.

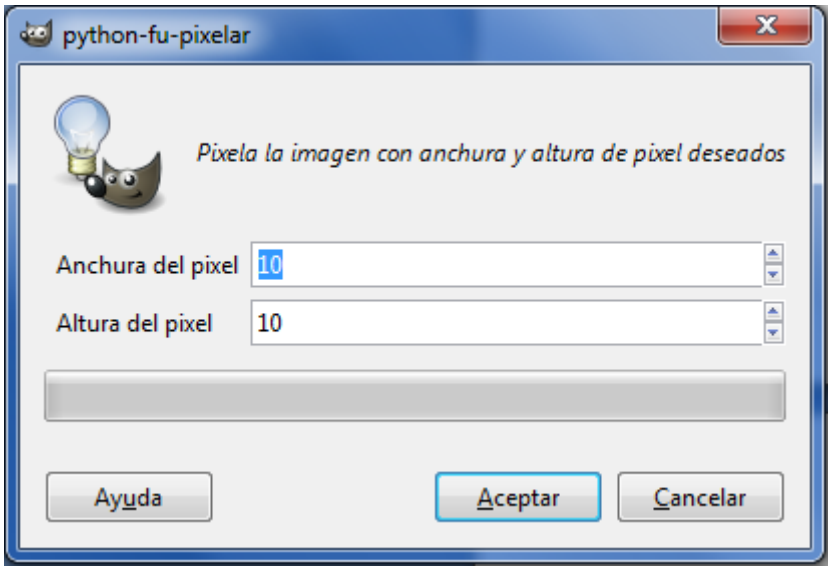

Figura 20. Cuadro de diálogo del procedimiento "pixelar".

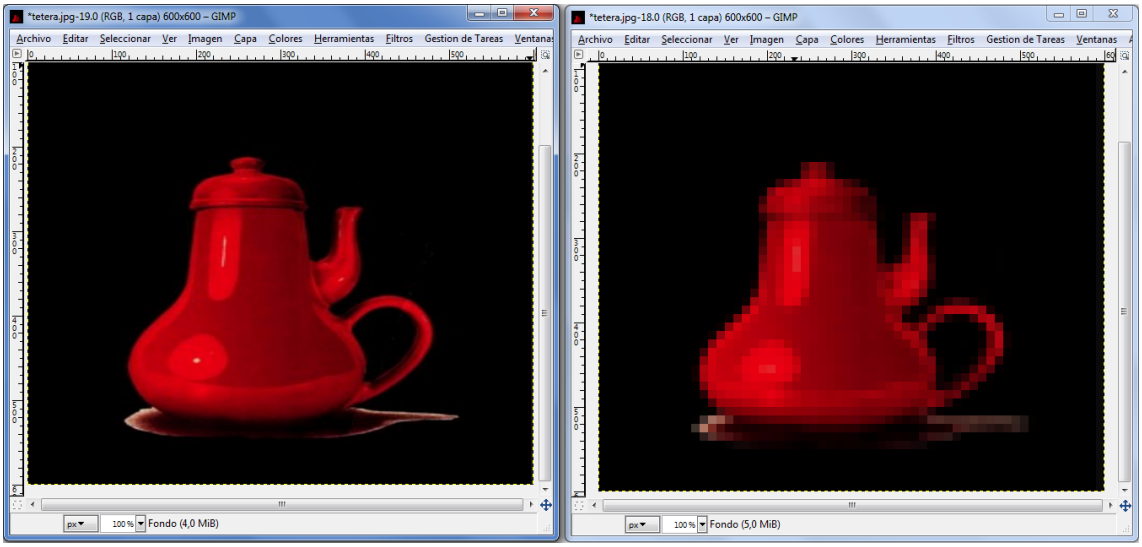

Figura 21. Antes y después de aplicar "pixelar".

-"Rotar".- Con este procedimiento se puede rotar una imagen. Se puede seleccionar el ángulo de rotación, el origen de la misma y la forma de recortar el resultado.

A continuación se muestra un ejemplo del funcionamiento de este procedimiento.

Figura 22. Efecto conseguido al aplicar "rotar".

-"Texto".- Se usa para insertar un texto en la imagen, determinando el color, la fuente del texto y la posición.

-"Tono y Saturacion".- Modifica el tono, luminosidad y saturación de una imagen.

Los procedimientos pueden ser usados en cualquier momento, pero si se usan sin crear una macro se produce el siguiente mensaje de

aviso: "No se ha iniciado el proceso. No se guardara esta accion.".

4.4.2 Grabar una nueva macro

Con la operación de grabar una nueva macro se crea un fichero XML que contiene la información referente a todos los procedimientos aplicados a la imagen.

Primero se inicia el proceso accediendo al menú "Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro".

A continuación se ejecutan los procedimientos que se desee en el menú "Gestion de Tareas/Procedimientos/...", tal y como se explica en el apartado 4.4.1 La macro que se crea contiene tantos procedimientos como se han usado, sin límite máximo.

Y por último en el menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro" se finaliza la grabación de la macro. Se introduce el nombre y ubicación del fichero XML que se desea que contenga la macro, y también el nombre y ubicación del fichero CSV de recomendaciones donde se desea que se almacenen el nombre de la macro, junto a las características de la imagen modificada.

Una vez terminada la operación, el usuario obtiene la imagen modificada con los procedimientos elegidos, y el fichero XML correspondiente a la macro creada, además de la ampliación del fichero de recomendaciones elegido.

4.4.3 *Ejecutar una macro*

Esta es la operación indicada para modificar una imagen con las macros creadas con anterioridad.

Utilizando el menú "Gestion de tareas/Acciones/Ejecutar macros", se introduce el nombre y la ubicación del fichero XML que corresponde a la macro que se desea aplicar. Entonces la aplicación abre una nueva ventana, con el nombre de la macro, en la que aparece la imagen modificada por la aplicación de los procedimientos de la macro elegida.

4.4.4 *Recomendación automática de macro*

El usuario puede solicitar de la aplicación una recomendación automática para modificar una imagen a partir de los valores de un fichero de recomendaciones CSV.

La ejecución de esta funcionalidad se inicia a través del menú "Gestion de tareas/Acciones/Recomendacion Automatica de macros". En la Figura 23 se muestra el cuadro de diálogo con las distintas opciones que ofrece.

Figura 23. Cuadro de dialogo del *plug-in* "recomendacion.py"

El usuario debe introducir el nombre y ubicación del fichero CSV de recomendaciones que va a utilizar.

También elige el valor de K, que indica la cantidad de recomendaciones que desea obtener.

Y por ùltimo debe seleccionar uno de los tres modos que se le ofrecen.

El modo "K-minimos" devuelve las K recomendaciones más ajustadas a los valores de la imagen actual; es decir, recomienda las macros que se han usado anteriormente con las K-imágenes más parecidas a la imagen actual. Además de devolver las imágenes modificadas con las macros recomendadas, muestra un mensaje indicando dichas recomendaciones, como se observa en la Figura 24.

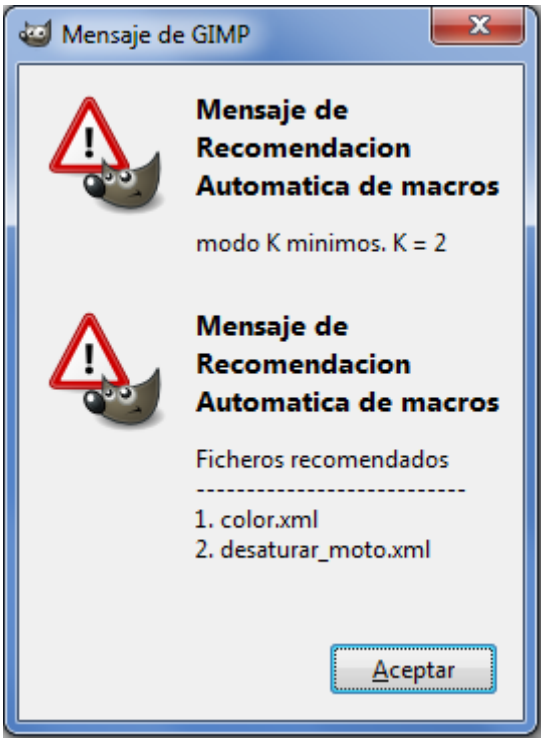

Figura 24. Mensaje de recomendaciones.

El modo "Dispersion" devuelve K recomendaciones dispersas de entre todas las recomendaciones posibles; es decir, si K=3 el programa devuelve la primera recomendación (que es la usada con la imagen más parecida), la última (que es la usada con la más diferente de las imágenes) y la intermedia, justo a la mitad.

Y por último el modo "Aleatorio" devuelve K recomendaciones distintas elegidas aleatoriamente de entre todas las que contienen el fichero CSV de recomendaciones.

Después de todo ello el usuario obtiene un mensaje con los nombres de las macros recomendadas, y también se abrirán tantas ventanas nuevas como recomendaciones se han solicitado (valor de K), conteniendo cada una de estas ventanas la imagen con el resultado de aplicar las macros recomendadas. Dichas imágenes nuevas toman por defecto el nombre de la macro usada.

4.5 *Manual de referencia*

Gracias a que GIMP es completamente extensible, como ya ha sido comentado anteriormente, trabajar en la mejora o ampliación de esta aplicación es bastante sencillo. A continuación se dan las instrucciones para crear más extensiones con *Python*, de igual manera que se han creado para esta aplicación. Se pueden crear extensiones con otros lenguajes, aunque la manera de hacerlo escapa al objetivo de este documento.

Lo primero que se debe hacer para trabajar con *Python* es instalar GIMP y el resto de elementos necesarios tal y como se indica en el apartado 10.1 de este mismo documento.

Para crear una nueva funcionalidad con un *plug-in,* simplemente se crea el fichero deseado con la extensión "`py`", y se ha de respetar la estructura descrita en el apartado 4.3.1 de este mismo documento. Este fichero ha de colocarse en la ubicación creada por GIMP llamada `\.gimp-2.6\plug-ins\`. Dependiendo del sistema operativo usado, esta carpeta estará ubicada en un sitio diferente. Al poner el fichero en esta ubicación y ejecutar GIMP, la nueva funcionalidad aparece en el menú indicado dentro del *plug-in*, siempre y cuando el código esté correcto en cuanto a sintaxis se refiere. Se deben respetar ciertas condiciones a la hora de crear una nueva funcionalidad. El nombre del *plug-in* no debe ser igual a ningún otro ya escrito anteriormente. El nombre de la función también ha de ser único. Hay que tener en cuenta el menú en el que se quiere introducir dicha aplicación, tal y como también se indica en el apartado 4.3.1.

La forma de depurar el código es directa; no existe compilador. Se edita el código, y se comprueba su funcionamiento directamente en GIMP.

Si se quiere extender este trabajo, hay que tener en cuenta los dos tipos de ficheros con los que interactúa la aplicación. *Python* los maneja como ficheros normales de texto. Los tipos de estos ficheros son XML los que contienen las macros y CSV para las recomendaciones, ambos formatos están muy extendidos en su utilización y han sido ampliamente descritos en este documento.

5. *Resultados*

En este capítulo se expone todo lo relacionado con la comprobación del funcionamiento final de la aplicación. La fase de pruebas se lleva a cabo simultáneamente a la fase de desarrollo y continúa también después de ella.

Las pruebas que se efectúan se pueden dividir en pruebas de verificación, pruebas de validación y las pruebas de eficiencia y de comportamiento de la aplicación.

Las pruebas de verificación se realizan mientras se desarrolla el código, y permiten detectar y corregir problemas en tiempo de ejecución. El tipo de errores detectados pueden conducir a que no se ejecute el elemento o que directamente se produzca alguna parada critica de la aplicación o del sistema.

Después se hacen las pruebas de validación, que consisten en comprobar que lo que hace cada elemento es lo que se espera que haga. En estas pruebas ha de quedar constatado el funcionamiento de todos los procedimientos aplicando la funcionalidad de cada uno de ellos sobre una imagen y observando el resultado. También se comprueba que los ficheros temporales, XML y CSV, son debidamente creados y las interacciones con ellos son correctas tanto para modificarlos como para leerlos.

Hay que destacar que todos los objetivos que persigue este proyecto son plasmados en dos tipos de ficheros muy versátiles y estándares, lo cual debe ser considerado como ventajoso, ya que cualquier cosa hecha en este proyecto puede ser adaptada en otros muchos sistemas de tratamiento de información. Esto puede ayudar a que esta aplicación no sea un sistema cerrado y aislado, ya que hay múltiples aplicaciones capaces de trabajar con los ficheros XML y

CSV. Y por supuesto este aspecto es muy usado en las comprobaciones de las que habla este capítulo.

5.1 *Pruebas de validación*

A continuación se muestra una secuencia de acciones para llevar a cabo la prueba de la aplicación completa.

- Abrir una imagen en GIMP.

Prueba de generación de macros.

- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro"
  - ✓ Comprobar que se han creado los ficheros temporales "archivo.tmp.xml" y "almacen.tmp.txt" en la ubicación ".\usuario\Mis documentos\Mis imágenes"
- Ejecutar un procedimiento cualquiera en el menú "Gestion de Tareas/Procedimientos/..."
- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro". En el cuadro de diálogo introducir el nombre de un nuevo fichero XML para la macro, y otro para el fichero de recomendaciones CSV.
  - ✓ Comprobar que ya no están los ficheros temporales "archivo.tmp.xml" y "almacen.tmp.txt" en la ubicación ".\usuario\Mis documentos\Mis imágenes".
  - ✓ Comprobar que se han creado correctamente los nuevos ficheros XML de la macro y CSV de recomendaciones en las ubicaciones indicadas.

Prueba de todos los procedimientos.

- Ejecutar un procedimiento cualquiera en el menú "Gestion de Tareas/Procedimientos/..."
  - ✓ Comprobar que se aplica su funcionalidad a la imagen.
  - ✓ Comprobar que aparece el mensaje "No se ha iniciado el proceso.No se guardara esta accion." Que indica que no se está grabando una nueva macro.

Repetir el paso anterior para todos los procedimientos.

Prueba de creación de macros con todos los procedimientos.

- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro"
- Ejecutar un procedimiento cualquiera en el menú "Gestion de Tareas/Procedimientos/..."
- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro". En el cuadro de diálogo introducir el nombre de un nuevo fichero XML para la macro, y seleccionar el fichero de recomendaciones CSV creado anteriormente.
  - ✓ Comprobar que se ha creado correctamente el nuevo fichero XML de la macro y que el fichero CSV de recomendaciones contienen la nueva recomendación.

Repetir el paso anterior para todos los procedimientos.

Prueba de excepción por falta de fichero `'archivo.tmp.xml'`.

- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro". En el cuadro de diálogo introducir el nombre de un nuevo fichero XML para la macro, y seleccionar el fichero de recomendaciones CSV creado anteriormente.

- ✓ Comprobar que aparece el mensaje "El archivo 'archivo.tmp.xml' no existe".

Prueba de excepción por falta de fichero 'almacen.tmp.txt'.

- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro"
- Eliminar el fichero "almacen.tmp.txt" que se ha creado en la ubicación ".\usuario\Mis documentos\Mis imágenes"
- Ir al menú "Gestion de tareas/Acciones/Grabacion de macros/Fin de grabación de macro". En el cuadro de diálogo introducir el nombre de un nuevo fichero XML para la macro, y seleccionar el fichero de recomendaciones CSV creado anteriormente.
  - ✓ Comprobar que aparece el mensaje "El fichero 'almacen.tmp.txt' no existe"

Prueba de ejecución de macros.

- Ir al menú "Gestion de tareas/Acciones/Ejecutar macros". En el cuadro de diálogo introducir el nombre de uno de los ficheros XML de las macros creadas anteriormente.
  - ✓ Comprobar que se abre una nueva ventana con la imagen modificada por la funcionalidad correspondiente.

Repetir el paso anterior para todas las macros creadas en pasos anteriores.

Pruebas de recomendación automática de macros.

- Solicitar la recomendación automática en el menú "Gestion de tareas/Acciones/Recomendacion Automatica de macros". Seleccionar el fichero de recomendaciones CSV, elegir el valor de K y seleccionar el modo "K minimos".

- ✓ Comprobar que se abren K distintas nuevas ventanas con las imágenes modificadas según las macros usadas.

- Ir al menú "Gestion de tareas/Acciones/Recomendacion Automatica de macros". En el cuadro de dialogo seleccionar el fichero de recomendaciones CSV, elegir el valor de K y seleccionar el modo "Dispersion".
  - ✓ Comprobar que se abren K distintas nuevas ventanas con las imágenes modificadas según las macros usadas.

Repetir el paso anterior para distintos valores de K.

- Ir al menú "Gestion de tareas/Acciones/Recomendacion Automatica de macros". En el cuadro de dialogo seleccionar el fichero de recomendaciones CSV, elegir el valor de K y seleccionar el modo "Aleatorio".
  - ✓ Comprobar que se abren K distintas nuevas ventanas con las imágenes modificadas según las macros usadas.

Repetir el paso anterior para distintos valores de K.

5.2 *Prueba del cálculo de la distancia euclidiana.*

El método usado para comprobar que el programa calcula correctamente la distancia euclidiana, se hace aprovechando que el fichero de recomendaciones CSV, que contiene los datos de los histogramas de las imágenes con las que se hacen las macros, es perfectamente editable en *Excel*.

Esta prueba consiste en aplicar la misma fórmula usada en el *plug-in* $\sum_{i=0}^{n} \frac{1}{M_i^2} \times (datos_i - actual_i)^2$, con los mismos datos, pero en Excel.

Se necesitan los datos del histograma de la imagen que es objeto de la recomendación, para manejarlos junto con todos los demás datos que ya están en el fichero CSV de recomendaciones. Para recoger estos valores del histograma de la imagen a tratar en el mismo formato que están guardados en un fichero CSV de recomendaciones, se va al menú "`Gestion de tareas/Acciones/Grabacion de macros/Inicio de grabación de macro`", y una vez hecho esto se edita el fichero temporal "`almacen.tmp.txt`" que se ha generado en la ubicación "`.\usuario\Mis documentos\Mis imágenes`", del cual se extraen dichos datos. Ahora que ya están todos los datos en el mismo fichero editable con *Excel*, se puede hacer los cálculos necesarios gracias a la potencia de *Excel*, y comparar los resultados.

5.3 *Pruebas de funcionamiento.*

A continuación se presentan varios ejemplos del funcionamiento real de la aplicación.

Primero se solicita una sola recomendación de la imagen más parecida. Se pide K=1 y el modo K-minimos.

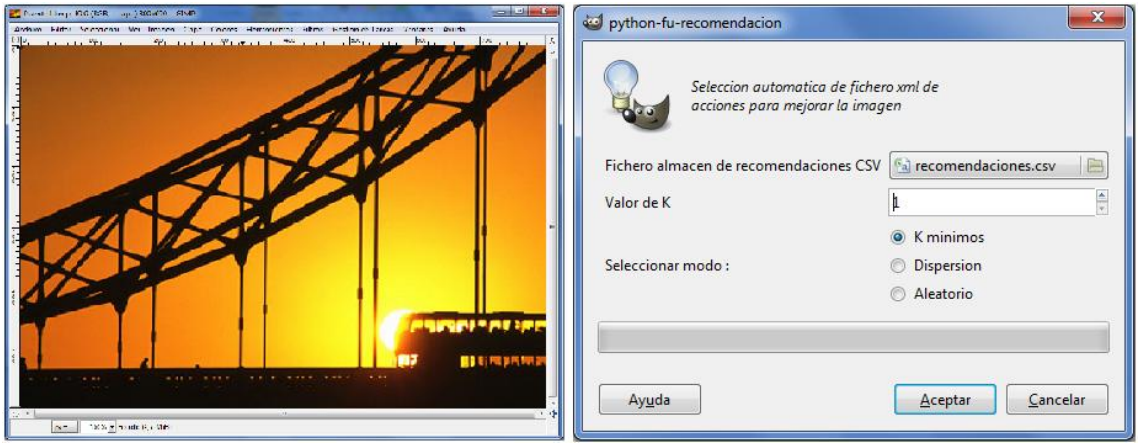

Figura 25. Petición de recomendación en modo `K-minimos`.

La aplicación devuelve un mensaje con el nombre de la macro recomendada y abre una nueva imagen la funcionalidad de la macro aplicada.

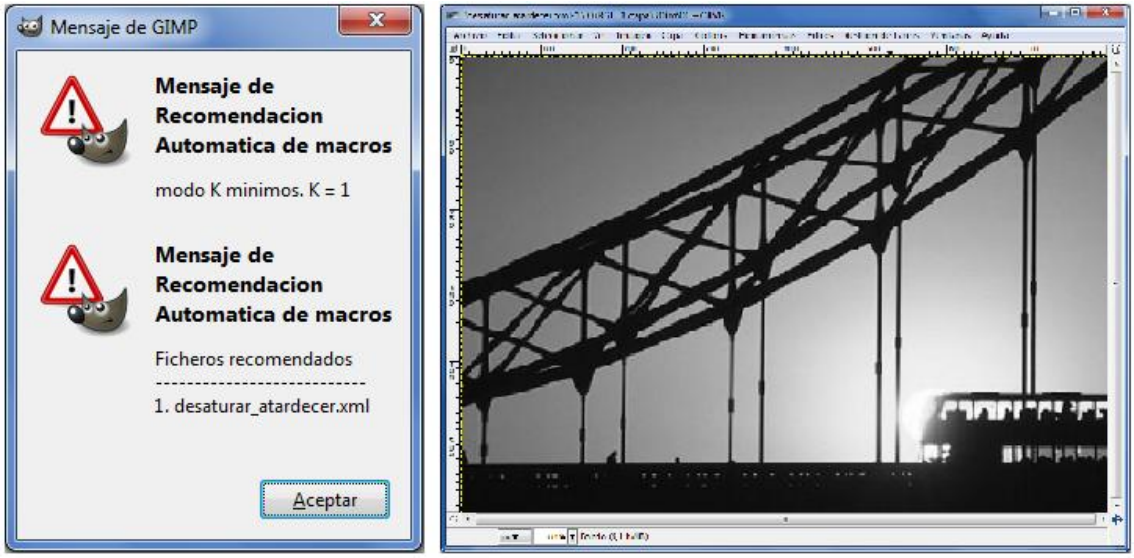

Figura 26. Mensaje de recomendación y la imagen retocada.

Se busca la macro que ha sido recomendada en el fichero de recomendaciones CSV.

```
"Pirámides.BMP";145.895;97.872;170;293;45;8;38;87;93;85;347;634;156;78;42;24;
18;16;27;998;0;0;0;0;0;0;1;"desaturar_atardecer.xml";
```

Se obtiene la imagen con la que se creó la macro, y sus valores del histograma.

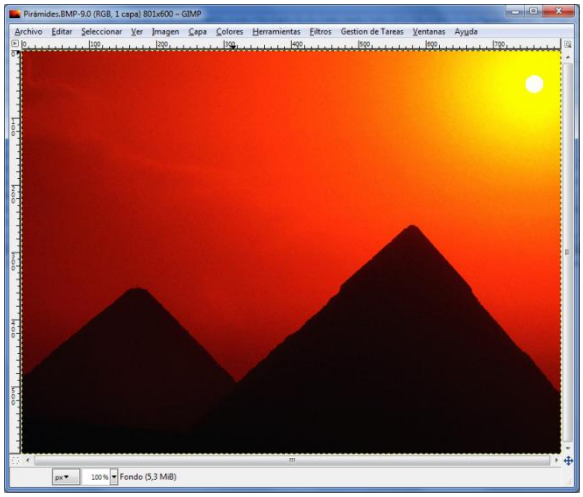

Figura 27. Imagen "Pirámides.BMP".

Este es el resultado de una solicitud de recomendaciones en modo "dispersión" con K=3.

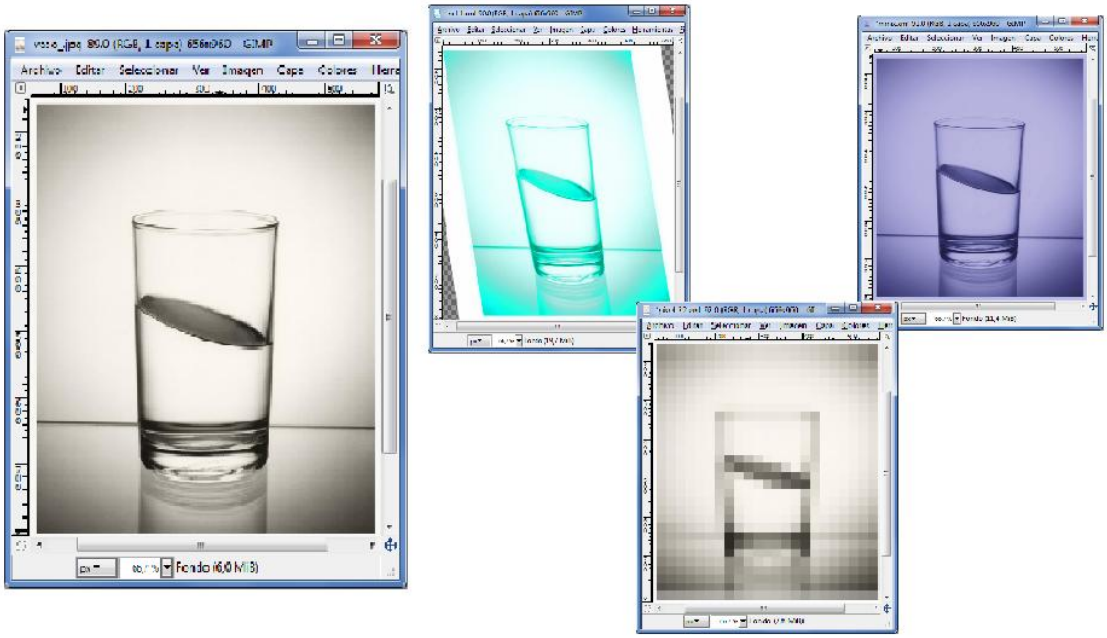

Figura 28. Recomendaciones en modo "dispersión" y K=3.

El siguiente ejemplo muestra el resultado de una petición de recomendaciones en modo "aleatorio" con K=3.

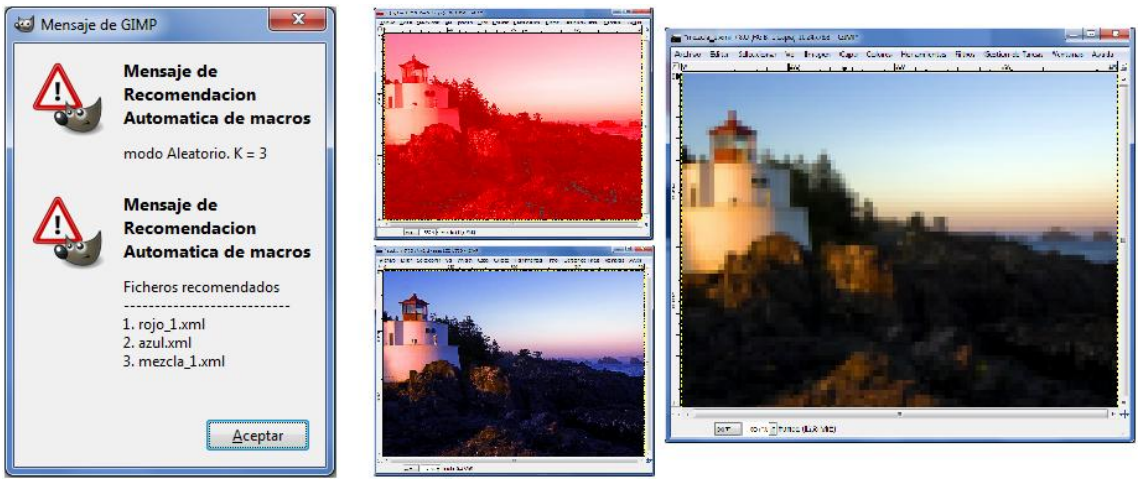

Figura 29. Recomendaciones en modo "aleatorio" y K=3.

A continuación se presenta una imagen junto al resultado de aplicar sobre ella una macro.

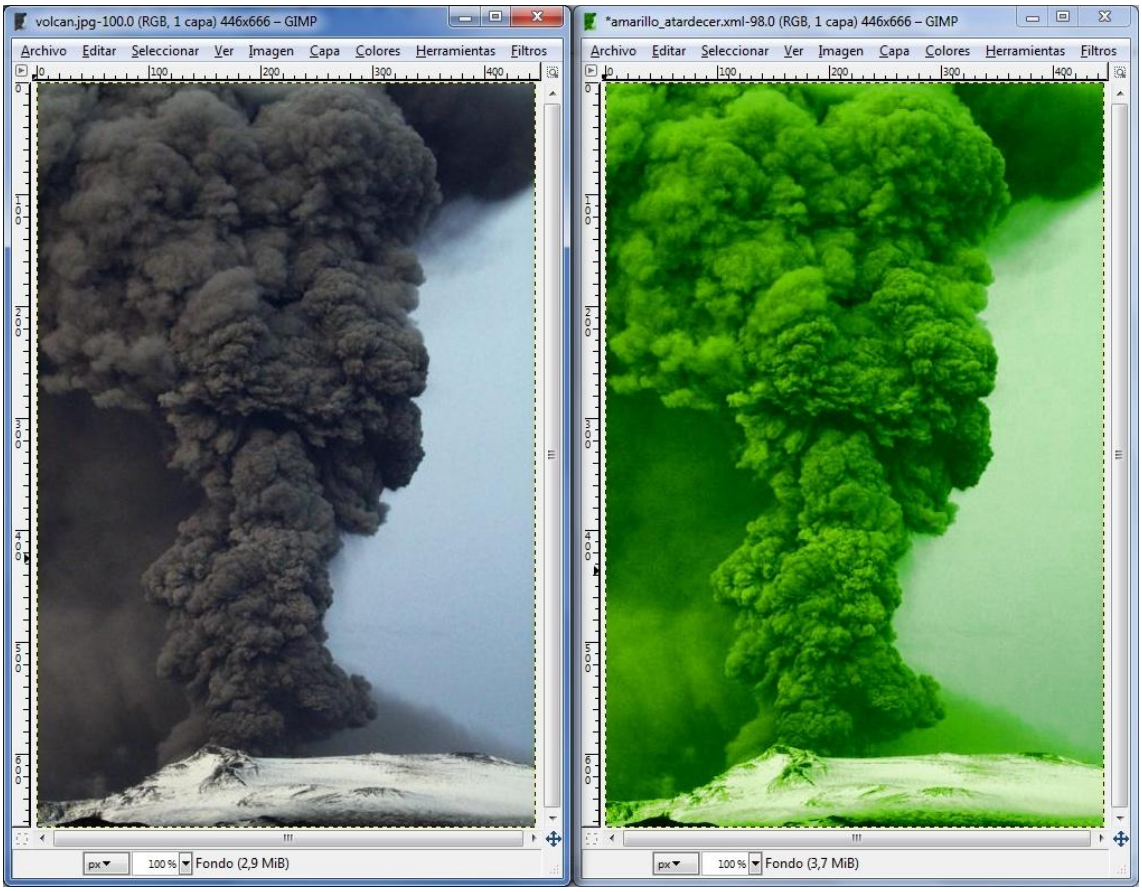

Figura 30. Imagen original (izquierda), y la imagen retocada.

6. *Gestión del proyecto*

La realización de este proyecto puede ser dividida claramente en tres fases. Una fase de investigación y estudio de los objetivos del proyecto, una fase central en la que se ha desarrollado la aplicación, y la fase final de documentación y prueba de la aplicación.

En la primera fase del proyecto se ha estudiado GIMP y las posibles formas de afrontar el proyecto trabajando con GIMP. Primero conocer GIMP, su uso y funcionamiento, y las posibles formas de hacerlo extensible. A la hora de afrontar el problema a resolver, hay que estudiar las distintas posibilidades que se ofrecen, para ver si eran o no válidas. Se puede decir que esta fase de trabajo va desde el momento del inicio del proyecto, hasta el momento en el que se llega a la conclusión de qué elementos usar para desarrollar la aplicación. El trabajo desarrollado en esta fase ha sido de 60 horas.

La siguiente fase comienza cuando se llega a la conclusión de usar complementos escritos en *Python* para el desarrollo de la aplicación. Este trabajo consiste en la elaboración del código y el desarrollo de la aplicación. Esta fase tiene una carga de trabajo de 80 horas.

El siguiente proceso de documentación es el que da como resultado el presente documento, y la carga de trabajo ha sido de 30 horas.

En cuanto a las necesidades materiales para realizar este proyecto, se ha usado un PC convencional, con unos requisitos muy básicos, por lo que el coste para el equipo de trabajo se puede encontrar en 500€.

En cuanto al software necesario se estima libre de coste, ya que se ha trabajado con software libre y el sistema operativo utilizado ha

sido *Windows* cuya licencia está incluida en el precio de compra del PC.

Por lo tanto a los costes de materiales se le suman los costes por el trabajo de un Ingeniero durante 150 horas, con un coste de 20€ la hora.

Material informático hardware……500€

Material informático Software…….0€

Mano de obra (150 hrs * 20€)……3.000€

Total……..3.500€

7. *Conclusiones*

La aplicación creada en este proyecto es una interesante herramienta destinada a ofrecer resultados automáticos e independientes del usuario. La idea sugiere que la aplicación va a "pensar" la acción de retoque fotográfico más apropiada para utilizar con una imagen. Sin embargo no se puede decir que la aplicación use inteligencia para tomar dicha decisión. La recomendación se basa en la experiencia almacenada en forma de macros de uno o varios usuarios, y la aplicación simplemente realiza una tarea de comparación entre todas las recomendaciones posibles para elegir una de ellas en función de unos criterios dados por el usuario.

Al tratarse de un tema subjetivo como es la decisión de qué acción es la más adecuada para retocar una imagen, se observa que no se trata de que la recomendación sea la objetivamente mejor para la imagen en cuestión, sino que es la recomendación basada en el actuación personal del usuario para retocar una imagen que tenía unos valores muy similares. Así pues, un fichero de recomendaciones de un usuario concreto, garantiza que su uso va a seguir los criterios de modificación de dicho usuario, o dicho de otra forma, se va a retocar la imagen en función de la experiencia particular del usuario que aporta el fichero de recomendaciones y se van a seguir sus criterios de estilo. Por esto mismo un fichero de recomendaciones en sí mismo puede ser muy valorado, como si fuese una aplicación o utilidad.

Una ventaja que ofrece esta aplicación es la de poder guardar toda la experiencia de un usuario adquirida con el trabajo de retoque fotográfico en macros, y poder recuperarla y usarla en cualquier momento. Cuando la modificación de un tipo de imagen requiere no solo la utilización de una herramienta aislada, sino que se requiere la

combinación de varias herramientas, esta aplicación puede ser muy útil. Si un usuario sabe lo que se quiere realizar exactamente para un determinado tipo de fotografía, y lo tiene convenientemente guardado en una macro, sin duda es una ayuda en su trabajo. Y esa misma ventaja la puede tener cualquier aficionado, o usuario que cuente con los ficheros de macros de otro usuario avanzado o profesional del retoque fotográfico.

Resulta interesante el trabajo realizado con los datos de los histogramas de las imágenes y el cálculo de la distancia euclidiana para ver la similitud entre ellos. Este aspecto deja entrever la posibilidad de aplicar la distancia euclidiana en otros campos, buscando la forma de caracterizar cualquier elemento con valores numéricos para poder compararlos después.

También hay que indicar aquí que gracias al carácter extensible de la aplicación, queda abierta a todo tipo de ampliación. Sin olvidar la ventaja que supone trabajar con formatos de ficheros tan reconocibles y estandarizados.

8. *Futuras líneas de trabajo*

Un primer punto de mejora de este trabajo es añadir todos los procedimientos posibles. Cuantas más funcionalidades se puedan usar para crear macros, más potente será la aplicación. Esta línea de trabajo es simple continuación de lo hecho hasta ahora, que consiste en encontrar procedimientos de GIMP, y crear el *plug-in* correspondiente que incluya el código para generar el fichero XML de la macro, y también que acumule la información en un fichero CSV de recomendaciones. También se debe incluir el código necesario en el *plug-in* `"macros.py"` encargado de la ejecución de macros, para que sea capaz de reconocer y ejecutar las nuevas macros creadas.

Otro punto de mejora de la aplicación, en cuanto a su eficiencia, puede ser el estudio del comportamiento de la fórmula usada para el cálculo de la distancia euclidiana. La fórmula utilizada puede ser modificada desde muchos aspectos, cantidad de sumandos elegidos, coeficientes para la normalización, etc.…

También, se puede buscar la utilidad del cálculo de la distancia euclidiana con los datos numéricos de otros tipos de elementos además de las imágenes.

En la fase inicial del proyecto se estudió la posibilidad de usar para la consecución de los objetivos del proyecto una herramienta de metadatos (ficheros XMP), que aunque quedó desestimada, puede ser una posible línea de trabajo. Esta herramienta permite trabajar con metadatos que se asocian a cada fichero de imagen. El origen de estos datos puede ser cualquiera, desde la propia cámara de fotos que crea el fichero de imagen, hasta cualquier retoque que se haga después o la inclusión de datos directamente. Es una forma de poder

tener muchos datos de una imagen, que pueden ser de mucha utilidad.

El uso de esta aplicación va a generar un buen número de ficheros XML de macros, y también de ficheros CSV de recomendaciones. Gracias a la gran compatibilidad de estos formatos, se puede estudiar otras utilidades de los datos que contienen.

9. *Bibliografía*

Andrés Marzal, I. G. (2003). *Introducción a la programación con Python.*

Duque, R. G. *Python para todos.*

E.Villate, J. (2001). *Introducción al XML.*

Gimp. (s.f.). Recuperado el 2011, de http://www.gimp.org/

http://www.corel.com. (s.f.).

León, J. M. (2008). *El proyecto GIMP.*

Medina, R. C. (2005). *Inmersión en Python.*

Pacheco, F. J. (2006). *Python-fu para no programadores. Creando scripts en Gimp.*

10. *Anexos*

10.1 *Instalación y configuración del sistema*

Mac OS X o sistemas operativos basados en *Unix*, están preparados con las herramientas necesarias para trabajar en GIMP con *Python.* Pero en *Windows* no es así, por lo que es necesario configurarlo. Seguramente hay otras formas de configurar *Windows* para trabajar en GIMP con *Python*, pero a continuación se describe la que se ha usado en este proyecto, especificando los elementos necesarios y enlaces donde descargarlos. Este método ha sido probado en distintas versiones de *Windows*.

Si el sistema tiene instalado GIMP y no *Python* hay que empezar por desinstalar GIMP.

En primer lugar hay que instalar *Python* en la maquina y después hay que reiniciarla. La versión que se ha usado en este proyecto ha sido *Python*-2.5.2.

A continuación hay que descargar e instalar los distintos elementos en el siguiente orden:

-La biblioteca para el entorno grafico (botones, ventanas, etc...) (Gtk-2.12.19 GTK+Runtime Environment) disponible en:

http://garr.dl.sourceforge.net/sourceforge/gladewin32/

-La biblioteca gráfica (pycairo-1.4.12-1) disponible en:

http://ftp.gnome.org/pub/GNOME/binaries/win32/pycairo/1.4/

- El binding[13] de la biblioteca gráfica GTK para *python* (pygobject-2.14.1-1) disponible en:

http://ftp.gnome.org/pub/GNOME/binaries/win32/pygobject/2.14/

-El binding de la biblioteca gráfica GTK para *python* (Pygtk-2.12.1-2) disponible en:

http://ftp.gnome.org/pub/GNOME/binaries/win32/pygtk/2.12/

Una vez hecho todo lo anterior hay que volver a reiniciar la maquina, y después instalar el GIMP. La versión usada en este proyecto ha sido GIMP 2.6.8.

[13] Un binding es una adaptación de una biblioteca para ser usada en un lenguaje distinto de aquel en el que ha sido escrita.

10.2 *Ficheros de la aplicación*

Fin.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in abre el fichero temporal "archivo.tmp.xml"en modo lectura y copia todo su
contenido en el fichero .xml definitivo que elige el usuario.
# definimos las funciones necesarias
def finalizacion(img, drawable,filename,filestore):
	#Apertura del fichero 'archivo.tmp.xml' para recuperar la informacion de las
acciones realizadas sobre la imagen.
	try:
		fichero_temporal = open("archivo.tmp.xml", "r")
	except:
		pdb.gimp_message("El archivo 'archivo.tmp.xml' no existe")
	else:
		#Apertura del fichero definitivo, elegido por el usuario , en el que se
guardaran las acciones realizadas.
		fichero_definitivo=open(filename,"w")

		#Se recorre el fichero temporal linea a linea y se copian en el
definitivo.
		while True:
			linea=fichero_temporal.readline()
			if not linea:break
			fichero_definitivo.write(linea)

		fichero_definitivo.write ("</procedimientos>")
		#Se cierran los ficheros.
		fichero_definitivo.close()
		fichero_temporal.close()

		#Con este tratamiento de excepcion, se evita crear un nuevo fichero si es
que no existe anteriormente.
		try:
			almacen=open(filestore,"r")
		except:
			pdb.gimp_message("El fichero de almacen no existe. Se creara
nuevo")
			new_file_alamcen=open(filestore,"w")
			new_file_alamcen.close()
		else:
			almacen.close()

		#Apertura del fichero temporal 'almacen.tmp.txt' para recuperar su
contenido y meterlo en el almacen definitivo.
		try:
			almacen_temporal=open("almacen.tmp.txt","r")
		except:
			pdb.gimp_message("El fichero 'almacen.tmp.txt' no existe")
		else:

			try:
				almacen=open(filestore,"a")
			except:
				pdb.gimp_message("El fichero almacen .csv no existe")
			else:
				linea_tmp=almacen_temporal.readline()
#*****************buscando el
separador*****************************************************************************
*****************************
				#Buscamos cual es el separador usado, ya que Windows usa
"\" y MAC usa "/"
				for caracter in filename:
					if (caracter==("\\"))|(caracter==("/")):
						separador=caracter
```

```
#******************buscando el
separador*****************************************************************************
*******************************
                                file_xml=filename.split(separador)#Separador es / o \ segun
estemos en MAC o Windows.
                                #Linea definitiva almacenada.Informacion del xml. : nombre
del fichero de imagen ; media ; Desviacion Estandar ; Mediana ; Pixeles en el rango(8
rangos de 32 niveles por cada color rojo, verde, azul) ; Nombre del fichero xml ;
                                almacen.write(linea_tmp+"\""+str(file_xml[-
1])+"\""+";"+"\n")
                                #Se cierran los ficheros , y se eliminan los ficheros
temporales "almacen.tmp.txt"  y "archivo.tmp.xml"
                                almacen.close()
                                almacen_temporal.close()
                                os.remove("almacen.tmp.txt")
                                os.remove("archivo.tmp.xml")

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "fin",
                "Finaliza el fichero xml que almacenara las acciones",
                "Finaliza el fichero xml que almacenara las acciones",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Acciones/Grabacion de macros/Fin de grabacion
de macro",
                "RGB*, GRAY*",
                [
                        (PF_FILENAME, "filename", "Nombre del fichero .xml", ""),
                        (PF_FILENAME, "filestore", "Fichero almacen de macros automaticas
.csv", "")
                ],
                [],
                finalizacion)
        main()
```

recomendación.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os,random

#Este plug-in ofrece una recomendacion de las acciones a efectuar sobre la imagen
actual.
# definimos las funciones necesarias
def recomendar(img, drawable,filestore,K,modo):
#**********declaracion de variables y de
listas*****************************************************************************
*********
        #Lista de los coeficientes usados para el calculo de la distancia euclidea. Hay
un coeficiente multiplicador para cada uno de los elementos usados en el
calculo.media,desv.est,Mediana,y el percentil de rojo x 8 rangos,verde x 8 rangos y azul
x 8 rangos.
        coeficientes=[]
        M=256.0
        for c in range(0,27):
                if c > 2 :
                        M=100.0
                p=1/(M**2)
                coeficientes.append(p)
#*****************************************************************************
**************
```

```
	recomendaciones=[]#Lista para almacenar las recomendaciones extraidas del fichero
almacen
	resultados_comparaciones=[]#Lista en la que se almacenan los resultados de la
distancia euclidiana de cada fichero xml con la imagen actual, y tambien se almacena el
nombre del fichero correspondiente.
	lista_xml=[]#Lista de los nombres de los ficheros xml recomendados
	color_i=[]#Lista que almacena los valores de pixeles en los niveles de los
colores RGB, en rangos de 8
	actual_i=[]#Lista de los valores del histograma de la imagen actual.
#**************************************************************************************
**************
#******************apertura y tratamiento del fichero almacen de
recomendaciones*****************************************************************************
*********
	#Se abre el fichero de recomendaciones elegido y se leen todas las lineas, y se
almacenan en la lista "recomendaciones"
	try:
		fichero_almacen=open(filestore,"r")
		while True:
			linea=fichero_almacen.readline()
			if not linea:break
			recomendaciones.append(linea)
		fichero_almacen.close()
	except:
		pdb.gimp_message("El archivo de almacen no existe")
	else:
#***************valores de la imagen actual a la lista
actual_i*************************************************************************************
******
		#Recopilacion de los datos estadisticos del histograma de la imagen para
la que se pide una recomendacion.Imagen actual.Los metemos en la lista actual_i
		histograma_actual=(pdb.gimp_histogram(drawable,0,0,255))

		for v in range(0,3):#valores de media,desv y Mediana
			actual_i.append("%.3f"%(histograma_actual[v]))
		#En la lista color_i se meten todos los valores de los distintos rangos
de los 3 colores.
		for color in range(0,3):#3 colores RGB
			color_i.append([])
			for rango in range(0,255,32):#8 rangos de 32 niveles
				pixel=pdb.gimp_histogram(drawable,color+1,rango,rango+31)
				color_i[color].append(int(pixel[5]*1000))
		#Pasamos los valores de la lista color_i a la lista actual_i
		for color_ in range(0,3):
			for rango_ in range(0,8):
				actual_i.append(color_i[color_][rango_])
#**************************************************************************************
**************
#**********valores de cada recomendacion a la
lista*****************************************************************************
********
		#Se recorren las lineas de la lista recomendaciones, y se extraen los
datos estadisticos de cada recomendacion para comparar con los de la imagen actual.
		for recomendacion in recomendaciones:
			datos=recomendacion.split(";") #Separa las cadenas entre las
";"....nombre fichero de la imagen ; media ; Desviacion Estandar ; Mediana ; Pixeles
totales ; Pixeles en el rango ; Percentil ; Nombre del fichero xml ;
			fichero_xml=(datos[-2])[1:len(datos[-2])-1]
#**********Calculo de la distancia euclidea y alamacenamiento del
resultado*************************************************************************************
************
			distancia_euclidea=0
			for d_e in range(0,27):

	distancia_euclidea=distancia_euclidea+(coeficientes[d_e]*(float(datos[d_e+1])-
(float(actual_i[d_e])))**2)
#*************P R U E B A
S*************************************************************************************
*
				#pdb.gimp_message("----------distancia euclidea -----------
----")
				#pdb.gimp_message(coeficientes[d_e])
				#pdb.gimp_message("*(")
				#pdb.gimp_message(float(datos[d_e+1]))
				#pdb.gimp_message(" - ")
				#pdb.gimp_message(float(actual_i[d_e]))
				#pdb.gimp_message(")**2 ===== ")
```

```
                                #pdb.gimp_message(coeficientes[d_e]*(float(datos[d_e+1])-
(float(actual_i[d_e])))**2)
#*************P R U E B A
S*************************************************************************************
*

        resultados_comparaciones.append([distancia_euclidea,fichero_xml])#se almacena el
resultado de cada comparacion y el fichero xml adjunto en una lista.
#*************************************************************************************
**************
#*************************************************************************************
**************
                #oredenacion de la lista 'resultados_comparaciones' por el metodo de la
burbuja, de menor a mayor.
                for i in range(1, len(resultados_comparaciones)):
                        for j in range(0, len(resultados_comparaciones)-i):
                                if (resultados_comparaciones[j][0]) >
(resultados_comparaciones[j+1][0]):
                                        elemento = (resultados_comparaciones[j])
                                        (resultados_comparaciones[j]) =
(resultados_comparaciones[j+1])
                                        (resultados_comparaciones[j+1]) = elemento
#*************************************************************************************
**************
                #Seleccion de los ficheros recomendados en funcion del 'modo' y de 'K'.
                tamano_recomendaciones=len(resultados_comparaciones)#Cantidad de
recomendaciones que tiene el fichero de recomendaciones
#*************P R U E B A
S*************************************************************************************
*
        #       pdb.gimp_message("-----------------resultados--------------")
        #       for b in resultados_comparaciones:
        #               pdb.gimp_message(b)
#*************************************************************************************
**************
                #--------------MODO K-minimos-----------------------------------------
---------
                if modo == "0":#Modo 'K minimos'. Se eligen los k ficheros con menor
distancia euclidea.
                        pdb.gimp_message("modo K minimos. K = "+str(int(K)))
                        for r in range(0, K):
                                #pdb.gimp_message(str(r+1)+" - El fichero xml (en modo 0)
recomendado es : "+resultados_comparaciones[r][1])
                                lista_xml.append(resultados_comparaciones[r][1])
                #--------------MODO Dispersion----------------------------------------
----------
                if modo == "1":#Modo 'Dispersion'. Se recomienda el primer y ultimo xml,
y los K-2 intermedios distanciados equidistantemente.
                        pdb.gimp_message("modo Dispersion. K = "+str(int(K)))
                        #Se eligen la primara recomendacion del total de la lista de
recomendaciones, y luego se cogen las sgte con el  ' intervalo=Total/K-1' ===>
'tamano_recomendaciones/(K-1)'
                        if K==1:#Se trata a parte por que en este caso K-1=0, y no
podriamos dividir por 0
                                pdb.gimp_message("K=1 ===> solo se muestra registro
central")
                                #pdb.gimp_message("1 - El fichero xml (en modo 1)
recomendado es : "+resultados_comparaciones[tamano_recomendaciones/2][1])

        lista_xml.append(resultados_comparaciones[tamano_recomendaciones/2][1])
                        else:
                                intervalo=int(tamano_recomendaciones/(K-1))
                                for r in range(0,K-1):
                                        #pdb.gimp_message(str((r*intervalo)+1)+" - El
fichero xml (en modo 1) recomendado es : "+resultados_comparaciones[r*intervalo][1])

        lista_xml.append(resultados_comparaciones[r*intervalo][1])
                                #pdb.gimp_message(str(tamano_recomendaciones)+" - El
fichero xml (en modo 1) recomendado es : "+resultados_comparaciones[-1][1])
                                lista_xml.append(resultados_comparaciones[-1][1])#Ultimo
xml de la lista, para cuando la division no es exacta.
                #--------------MODO Aleatorio-----------------------------------------
---------
                if modo == "2":#Modo 'Aleatorio'.Se eligen K ficheros al azar eliminando
los repetidos.
                        pdb.gimp_message("modo Aleatorio. K = "+str(int(K)))
                        while len(lista_xml) < K:
```

```
                                elegido=(random.choice(resultados_comparaciones))[1]
                                repetido=False
                                for l in lista_xml:
                                        if l == elegido:
                                                repetido=True
                                if repetido == False:
                                        lista_xml.append(elegido)
#*************************************************************************************
**************
#*****************buscando el
separador*****************************************************************************
****
                #Buscamos cual es el separador usado, ya que Windows usa "\" y MAC usa
"/"
                for caracter in filestore:
                        if (caracter==("\\"))|(caracter==("/")):
                                separador=caracter
#*****************buscando el
separador*****************************************************************************
******
                #Se ejecuta el plug-in 'macros' dandole como parametro el fichero
xml(direccion completa) que ha resultado para ejecutar.El fichero almace.txt ha de estar
en el directorio donde estan todos los xml
                f=filestore.split(separador)
                ficheros_recomendados="Ficheros recomendados \n-------------------------
-\n"
                for t in range(0,K):
                        ficheros_recomendados=(ficheros_recomendados+str(t+1)+".
"+lista_xml[t]+"\n")
                        path=(filestore[0:(len(filestore)-len(f[-1]))])+lista_xml[t]
                        pdb.python_fu_macros(img,drawable,path)
                pdb.gimp_message(ficheros_recomendados)#para sacar en un solo mensaje
todos los ficheros recomendados.

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "recomendacion",
                "Seleccion automatica de fichero xml de acciones para mejorar la imagen",
                "Seleccion automatica de fichero xml de acciones para mejorar la imagen",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Acciones/Recomendacion Automatica de macros",
                "RGB*, GRAY*",
                [
                        (PF_FILENAME, "filestore", "Fichero almacen de recomendaciones
CSV", ""),
                        (PF_SPINNER, "K", "Valor de K", 1, (1, 1000, 1)),
                        (PF_RADIO, "modo", "Seleccionar modo : ", "0", (("K minimos",
"0"),("Dispersion","1"),("Aleatorio","2")))
                ],
                [],
                recomendar)
        main()
```

macros.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os,re,pickle
```

```
# definimos las funciones necesarias
#Este plug-in permite al usuario elegir un fichero .xml que contienen una serie de
acciones realizadas sobre una foto, y hacer que replique dichas acciones sobre la foto
deseada.
def leer(imagen, drawable,filename):
	#Duplicamos la imagen, y hacemos los cambios para que las acciones se ejecuten
sobre la copia, tanto para la imagen, como para el drawable.
	img=pdb.gimp_image_duplicate(imagen)
	pdb.gimp_display_new(img)

	pdb.gimp_image_set_filename(img,filename)#Nombre que toma la nueva copia sobre la
que se realizan los cambios.

	drawable=pdb.gimp_image_get_active_drawable(img)
	a=-1
	v2=v3=v5=False
	linea=[]	#lista donde se almacenaran las lineas leidas del fichero .xml
	procedimientos=[]	#lista donde se alamacenaran los nombres de los
procedimientos y los valores de los parametros que usan

	#Se abre el fichero .xml elegido y se leen todas las lineas, y se almacenan en la
lista "linea"
	try:
		f=open(filename,"r")
		while True:
			l=f.readline()
			if not l:break
			linea.append(l)
		f.close()
	except:
		pdb.gimp_message("El archivo no existe")

	#Se recorren las lineas de la lista linea y se extraen los nombres de los
procedimientos que se usan y los valores de los parametros que necesitan
	for num_linea in linea:
		if re.search( "<procedimiento nombre=.+>",num_linea):#si la linea en
cuestion es en la que esta el nombre del procedimiento
			n_procedimiento=num_linea.split("\"") #Separa las cadenas entre
las comillas....
			procedimientos.append([n_procedimiento[1]])#..y almacena el nombre
del procedimiento que es el segundo (0,1,2,3..), en la lista "procedimientos"
			a=a+1
		if re.search("<parametro name =",num_linea):#De las lineas que tienen los
parametros...
			n_parametro=num_linea.split("\"") #..se selcciona el valor...
			procedimientos[a].append(n_parametro[3])#...y se almacena a
continuacion de los procedimientos

	#Recorre la lista "procedimientos" y hace la ejecucion de los procedimientos con
los parametros recuperados del fichero .xml elegido
	for fun in procedimientos:
		if fun[0]=="gimp_color_balance":
			if fun[2]=="True":#la variable es recuperada como un string, y hay
que pasarla como booleano (True/False)
				v2=True
			pdb.gimp_color_balance(drawable,fun[1],v2,fun[3],fun[4],fun[5])

		if fun[0]=="gimp_brightness_contrast":
			pdb.gimp_brightness_contrast(drawable,fun[1],fun[2])

		if fun[0]=="gimp_image_scale":
			pdb.gimp_image_scale(img,fun[1],fun[2])

		if fun[0]=="gimp_hue_saturation":
			pdb.gimp_hue_saturation(drawable,fun[1],fun[2],fun[3],fun[4])

		if fun[0]=="python_fu_fisheye":
			#fun[3]=fun[3].strip(" ()RGB")#Quitamos "RGB" y los parentesis
().strip no funciona en MAC
			fun[3]=(fun[3])[5:len(fun[3])-1]#Quitamos "RGB" y los parentesis
()
			#El valor de color_borde hay que recuperarlo como una tupla de
valores reales. En fun[3] tenemos la cadena [0,0,0,255].
			colores=(fun[3].split(","))#Creamos una lista de cadenas

	color=(float(colores[0]),float(colores[1]),float(colores[2]),float(colores[3]))#C
reamos la tupla con valores float
```

```
                        pdb.python_fu_fisheye(img,drawable,fun[1],fun[2],color)

                if fun[0]=="gimp_desaturate_full":
                        pdb.gimp_desaturate_full(drawable,fun[1])

                if fun[0]=="gimp_drawable_transform_rotate_default":
                        if fun[2]=="True":#la variable es recuperada como un string, y hay
que pasarla como booleano (True/False)
                                v2=True
                        if fun[5]=="True":#la variable es recuperada como un string, y hay
que pasarla como booleano (True/False)
                                v5=True

        pdb.gimp_drawable_transform_rotate_default(drawable,fun[1],v2,fun[3],fun[4],v5,fu
n[6])

                if fun[0]=="gimp_drawable_transform_shear_default":#inclinar
                        if fun[3]=="True":#la variable es recuperada como un string, y hay
que pasarla como booleano (True/False)
                                v3=True

        pdb.gimp_drawable_transform_shear_default(drawable,fun[1],fun[2],v3,fun[4])

                if fun[0]=="plug_in_red_eye_removal":
                        pdb.plug_in_red_eye_removal(img,drawable,fun[1])
                if fun[0]=="python_fu_texto":
                        fun[2]=fun[2].strip(" ()RGB")#Quitamos "RGB" y los parentesis ()
                        #El valor de color_borde hay que recuperarlo como una tupla de
valores reales. En fun[3] tenemos la cadena [0,0,0,255].
                        colores=(fun[2].split(","))#Creamos una lista de cadenas

        color=(float(colores[0]),float(colores[1]),float(colores[2]),float(colores[3]))#C
reamos la tupla con valores float
        pdb.python_fu_texto(img,drawable,fun[1],color,fun[3],fun[4],fun[5])
                if fun[0]=="plug_in_blur":
                        pdb.plug_in_blur(img,drawable)

                if fun[0]=="plug_in_pixelize2":
                        pdb.plug_in_pixelize2(img,drawable,fun[1],fun[2])

                if fun[0]=="plug_in_nlfilt":
                        pdb.plug_in_nlfilt(img,drawable,fun[1],fun[2],fun[3])

                if fun[0]=="gimp_levels_stretch":
                        pdb.gimp_levels_stretch(drawable)
# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "macros",
                "ejecuta la macro seleccionada",
                "ejecuta la macro seleccionada",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Acciones/Ejecutar macros",
                "RGB*, GRAY*",
                [
                (PF_FILENAME, "filename", "Nombre del fichero .xml", "")
        ],
                [],
                leer)
        main()
```

inicio.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os
```

```
#Este plug-in inicia una secuencia en la que el usuario podra almacenar las acciones que
efectue sobre una imagen en un fichero para poder recuperar en cualquier momento.
# definimos las funciones necesarias
def iniciacion(img, drawable):
	#Se inicia el archivo temporal 'archivo.tmp.xml', y se le introduce la cabecera
xml.
	file_tmp = open("archivo.tmp.xml", "w")

	file_tmp.write ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n\n")#cabecera xml
	file_tmp.write ("<procedimientos>\n")

	file_tmp.close()

	#Se inicia el archivo temporal 'almacen.tmp.txt', y se le introduce el nombre de
la imagen, y sus valores del histograma.
	almacen_tmp=open("almacen.tmp.txt","w")
	#Nombre del fichero de la imagen.
	nombre_file_imagen=pdb.gimp_image_get_name(img)
	almacen_tmp.write("\""+nombre_file_imagen+"\""+";")
	#Valores de histograma = media,desv.est y Mediana
	valores_histograma=pdb.gimp_histogram(drawable,0,0,255)
	almacen_tmp.write(str("%.3f"%(valores_histograma[0]))+";"+str("%.3f"%(valores_his
tograma[1]))+";"+str(int(valores_histograma[2]))+";")
	#Se guaradan los valores del histograma. 8 rangos de 32 niveles (8*32=256) para
cada color rojo,verde y azul*********************************
	for color in range(1,4):
		for rango in range(0,255,32):

	pixel=int((pdb.gimp_histogram(drawable,color,rango,rango+31))[5]*1000)
			almacen_tmp.write(str(pixel)+";")
	almacen_tmp.close()

# función principal
if __name__ == '__main__':

	# llamada a función register
	register(
		"inicio",
		"Iniciacion y apertura de archivo xml",
		"Iniciacion y apertura de archivo xml",
		"Jose Antonio",
		"Jose Antonio",
		"2011",
		"<Image>/Gestion de Tareas/Acciones/Grabacion de macros/Inicio de
grabacion de macro",
		"RGB*, GRAY*",
		[

		],
		[],
		iniciacion)
	main()
```

nivel_auto.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
```

```
import os

#Este plug-in permite desenfocar una imagen, y alamcena los datos de las acciones
efectuadas en un fichero .xml
# definimos las funciones necesarias
def nivel_automatico(img, drawable):
	#ejecucion de la accion
	pdb.gimp_levels_stretch(drawable)#Ejecucion del procedimiento .

	procedimiento="gimp_levels_stretch"# se guardan en variables el nombre del
procedimiento y los parametros usados.

	#Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
	try:
		file = open("archivo.tmp.xml", "r")
	except:
		pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
	else:
		file.close()

		#Se abre el fichero y se añaden los datos
		f = open("archivo.tmp.xml", "a")

		f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
		f.write ("\t\t<parametros>\n")
		f.write ("\t\t</parametros>\n")
		f.write ("\t</procedimiento>\n")

		f.close()

# función principal
if __name__ == '__main__':

	# llamada a función register
	register(
		"nivel_automatico",
		"Ajuste automatico de los niveles de color",
		"Ajuste automatico de los niveles de color",
		"Jose Antonio",
		"Jose Antonio",
		"2011",
		"<Image>/Gestion de Tareas/Procedimientos/Nivel Automatico de color",
		"RGB*, GRAY*",
		[
		],
		[],
		nivel_automatico)
	main()
```

filtro_NL.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in aplica el flitro NL sobre una imagen, y alamcena los datos de las acciones
efectuadas en un fichero .xml
# definimos las funciones necesarias
def filtro_NL(img, drawable,alpha,radius,filter):
	#ejecucion de la accion
	pdb.plug_in_nlfilt(img, drawable,alpha,radius,filter)#Ejecucion del
procedimiento.

	procedimiento="plug_in_nlfilt"# se guardan en variables el nombre del
procedimiento y los parametros usados.
```

```
        parametro1="alpha"
        parametro2="radius"
        parametro3="filter"
        v_parametro1=alpha
        v_parametro2=radius
        v_parametro3=filter

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "filtro_NL",
                "Aplica el filtro NL",
                "Aplica el filtro NL",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Filtro NL",
                "RGB*, GRAY*",
                [
                (PF_SLIDER, "alpha", "Alfa :", 0.00, (0.00, 1.00, 0.01)),
                (PF_SLIDER, "radius", "Radio :", 0.00, (0.00, 1.00, 0.01)),
                (PF_RADIO, "filter", "Filtro", "0", (("Media seccionada Alfa",
"0"),("Estimacion Optima","1"),("Realze de Bordes","2")))
                ],
                [],
                filtro_NL)
        main()
```

pixelizar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite pixelar una imagen eligiendo el ancho del pixel, y alamcena los
datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def pixeliza(img, drawable,pixel_width,pixel_height):
        #ejecucion de la accion
```

```
        pdb.plug_in_pixelize2(img,drawable,pixel_width,pixel_height)#Ejecucion del
procedimiento.

        procedimiento="plug_in_pixelize2"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="pixel_width"
        parametro2="pixel_height"
        v_parametro1=int(pixel_width)#este valor hay que guardarlo como integer
        v_parametro2=int(pixel_height)#este valor hay que guardarlo como integer

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "pixelar",
                "Pixela la imagen con anchura y altura de pixel deseados",
                "Pixela la imagen con anchura y altura de pixel deseados",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Pixelar",
                "RGB*, GRAY*",
                [
                        (PF_SPINNER, "pixel_width", "Anchura del pixel", 1, (1, 100, 1)),
                        (PF_SPINNER, "pixel_height", "Altura del pixel", 1, (1, 100, 1))
                ],
                [],
                pixeliza)
        main()
```

tono&Saturacion.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite hacer cambios en el tono y al saturacion de una imagen, y alamcena
los datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def tono_saturacion(img, drawable,hue_range,hue_offset,lightness,saturation):
        #ejecucion de la accion
```

```
        pdb.gimp_hue_saturation(drawable,hue_range,hue_offset,lightness,saturation)#Ejecu
cion del procedimiento.

        procedimiento="gimp_hue_saturation"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="hue_range"
        parametro2="hue_offset"
        parametro3="lightness"
        parametro4="saturation"
        v_parametro1=hue_range
        v_parametro2=hue_offset
        v_parametro3=lightness
        v_parametro4=saturation

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro4+"\" valor
=\""+str(v_parametro4)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "tono_saturacion",
                "Modifica tono, luminosidad y saturacion",
                "Modifica tono, luminosidad y saturacion",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Tono y Saturacion",
                "RGB*, GRAY*",
                [
                        (PF_RADIO, "hue_range", "Rango de tonos afectados", "0", (("Todos
los tonos", "0"),("Tonos rojos","1"),("Tonos amarillos","2"),("Tonos
verdes","3"),("Tonos cyan","4"),("Tonos azules","5"),("Tonos magenta","6"))),
                        (PF_SLIDER, "hue_offset", "Tono:", 0, (-180, 180, 1)),
                        (PF_SLIDER, "lightness", "Luminosidad:", 0, (-100, 100, 1)),
                        (PF_SLIDER, "saturation", "Saturacion:", 0, (-100, 100, 1))
                ],
                [],
                tono_saturacion)
        main()
```

add_texto.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-
```

```
# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite introducir texto en una imagen, y alamcena los datos de las
acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def add_texto(img, drawable,texto, color, fuente, posicion_x, posicion_y):
	#ejecucion de la accion
	pdb.python_fu_texto(img,drawable,texto, color, fuente, posicion_x,
posicion_y)#Ejecucion del procedimiento.

	procedimiento="python_fu_texto"# se guardan en variables el nombre del
procedimiento y los parametros usados.

	parametro1="texto"
	parametro2="color"
	parametro3="fuente"
	parametro4="posicion_x"
	parametro5="posicion_y"
	v_parametro1=texto
	v_parametro2=color
	v_parametro3=fuente
	v_parametro4=int(posicion_x)#este valor hay que guardarlo como integer
	v_parametro5=int(posicion_y)#este valor hay que guardarlo como integer

	#Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
	try:
		file = open("archivo.tmp.xml", "r")
	except:
		pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
	else:
		file.close()

		#Se abre el fichero y se añaden los datos
		f = open("archivo.tmp.xml", "a")

		f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
		f.write ("\t\t<parametros>\n")
		f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
		f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
		f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
		f.write ("\t\t\t<parametro name =\""+parametro4+"\" valor
=\""+str(v_parametro4)+"\"/>\n")
		f.write ("\t\t\t<parametro name =\""+parametro5+"\" valor
=\""+str(v_parametro5)+"\"/>\n")
		f.write ("\t\t</parametros>\n")
		f.write ("\t</procedimiento>\n")

		f.close()

# función principal
if __name__ == '__main__':

	# llamada a función register
	register(
		"add_texto",
		"Introduce texto en la imagen",
		"Introduce texto en la imagen",
		"Jose Antonio",
		"Jose Antonio",
		"2011",
		"<Image>/Gestion de Tareas/Procedimientos/Texto",
		"RGB*, GRAY*",
		[
			(PF_STRING, "texto", "Texto", "Colocar un texto"),
			(PF_COLOR, "color", "Color", (0,0,0)),
			(PF_FONT, "fuente","Fuente", "Sans 10"),
			(PF_SPINNER, "posicion_x", "Posicion X", 50, (0, 9999999999, 1)),
```

```
                        (PF_SPINNER, "posicion_y", "Posicion Y", 50, (0, 9999999999, 1)),
                ],
                [],
                add_texto)
        main()
```

brillo.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite hacer cambios en el brillo y contraste de una imagen, y alamcena
los datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def brillo_contraste(img, drawable,brightness,contrast):
        #ejecucion de la accion
        pdb.gimp_brightness_contrast(drawable,brightness,contrast)#Ejecucion del
procedimiento de cambio de brillo y de contraste

        procedimiento="gimp_brightness_contrast"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="brightness"
        parametro2="contrast"
        v_parametro1=int(brightness)#este valor hay que guardarlo como integer
        v_parametro2=int(contrast)#este valor hay que guardarlo como integer

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "brillo",
                "Ajuste del brillo y contraste",
                "Ajuste del brillo y contraste",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Cambiar brillo y contraste",
                "RGB*, GRAY*",
                [
                        (PF_SLIDER, "brightness", "Brillo", 0, (-127, 127, 1)),
```

```
                        (PF_SLIDER, "contrast", "Contraste", 0, (-127, 127, 1))
                ],
                [],
                brillo_contraste)
        main()
```

color.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite hacer cambios en el balance de color de una imagen, y alamcena los
datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def balance_color(img,
drawable,transfer_mode,conserva_luminosidad,cian_rojo,mag_verde,am_azul):
        #ejecucion de la accion
        pdb.gimp_color_balance(drawable,transfer_mode,conserva_luminosidad,cian_rojo,mag_
verde,am_azul)#Ejecucion del procedimiento de cambio de balance de color.

        procedimiento="gimp_color_balance"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="transfer_mode"
        parametro2="conserva_luminosidad"
        parametro3="cian_rojo"
        parametro4="mag_verde"
        parametro5="am_azul"
        v_parametro1=transfer_mode
        v_parametro2=conserva_luminosidad
        v_parametro3=cian_rojo
        v_parametro4=mag_verde
        v_parametro5=am_azul

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro4+"\" valor
=\""+str(v_parametro4)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro5+"\" valor
=\""+str(v_parametro5)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':
```

```
        # llamada a función register
        register(
                "color",
                "Balance de color",
                "Balance de color",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Balance de color",
                "RGB*, GRAY*",
                [
                        (PF_RADIO, "transfer_mode", "Seleccione el rango para ajustar",
"0", (("sombras", "0"),("tonos medios","1"),("puntos de luz","2"))),
                        (PF_TOGGLE, "conserva_luminosidad", "Conservar la luminosidad",
True),
                        (PF_SLIDER, "cian_rojo", "cian - rojo", 0, (-100, 100, 1)),
                        (PF_SLIDER, "mag_verde", "magenta - verde", 0, (-100, 100, 1)),
                        (PF_SLIDER, "am_azul", "amarillo - azul", 0, (-100, 100, 1))
                ],
                [],
                balance_color)
        main()
```

desenfocar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite desenfocar una imagen, y alamcena los datos de las acciones
efectuadas en un fichero .xml
# definimos las funciones necesarias
def desenfocar(img, drawable):
        #ejecucion de la accion
        pdb.plug_in_blur(img,drawable)#Ejecucion del procedimiento desaturate

        procedimiento="plug_in_blur"# se guardan en variables el nombre del procedimiento
y los parametros usados.

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "desenfocar",
                "Desenfoca una imagen",
```

```
                "Desenfoca una imagen",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Desenfocar",
                "RGB*, GRAY*",
                [
                ],
                [],
                desenfocar)
        main()
```

escalar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite cambiar el tamanio de una imagen, y alamcena los datos de las
acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def redimensionar(img, drawable,width, height):
        #ejecucion de la accion
        pdb.gimp_image_scale(img, width, height)#Ejecucion del procedimiento de escalado
de la imagen

        procedimiento="gimp_image_scale"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="width"
        parametro2="height"
        v_parametro1=int(width)#este valor hay que guardarlo como integer
        v_parametro2=int(height)#este valor hay que guardarlo como integer
        t_parametro1=type(width)
        t_parametro2=type(height)

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "escalar",
                "Escalar la imagen",
```

```
                "Escalar la imagen",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Escalar",
                "RGB*, GRAY*",
                [
                        (PF_SPINNER, "width", "Ancho", 0, (0, 1000, 1)),
                        (PF_SPINNER, "height", "Alto", 0, (0, 1000, 1))
                ],
                [],
                redimensionar)
        main()
```

ojos_rojos.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite quitar el efecto de ojos rojos  de una imagen provocado por el
flash, y alamcena los datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def ojos_rojos(img, drawable,threshold):
        #ejecucion de la accion
        pdb.plug_in_red_eye_removal(img,drawable,threshold)#Ejecucion del procedimiento.

        procedimiento="plug_in_red_eye_removal"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="threshold"
        v_parametro1=int(threshold)#este valor hay que guardarlo como integer.

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "ojos_rojos",
                "Quita los ojos rojos causado por los flashes de las camaras",
                "Quita los ojos rojos causado por los flashes de las camaras",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Ojos_rojos",
```

```
                "RGB*, GRAY*",
                [
                        (PF_SLIDER, "threshold", "Umbral", 0, (0, 100, 1))
                ],
                [],
                ojos_rojos)
        main()
```

rotar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite rotar una imagen, y alamcena los datos de las acciones efectuadas
en un fichero .xml
# definimos las funciones necesarias
def rotar(img, drawable,angulo,auto_center,center_x,center_y,interpolate,clip_result):
        pi=3.14159265358979323846264338327795
        angulo=((angulo*pi)/float(180))#Conversion del angulo de grados a radianes

        #ejecucion de la accion
        pdb.gimp_drawable_transform_rotate_default(drawable,
angulo,auto_center,center_x,center_y,interpolate,clip_result)#Ejecucion del
procedimiento de escalado de la imagen

        procedimiento="gimp_drawable_transform_rotate_default"# se guardan en variables
el nombre del procedimiento y los parametros usados.

        parametro1="angulo"
        parametro2="auto_center"
        parametro3="center_x"
        parametro4="center_y"
        parametro5="interpolate"
        parametro6="clip_result"
        v_parametro1=angulo
        v_parametro2=auto_center
        v_parametro3=int(center_x)#este valor hay que guardarlo como integer
        v_parametro4=int(center_y)#este valor hay que guardarlo como integer
        v_parametro5=interpolate
        v_parametro6=clip_result

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro4+"\" valor
=\""+str(v_parametro4)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro5+"\" valor
=\""+str(v_parametro5)+"\"/>\n")
```

```
                f.write ("\t\t\t<parametro name =\""+parametro6+"\" valor
=\""+str(v_parametro6)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "rotar",
                "Rotar la imagen",
                "Rotar la imagen",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Rotar",
                "RGB*, GRAY*",
                [
                        (PF_SLIDER, "angulo", "Angulo", 0, (-180, 180, 1)),
                        (PF_TOGGLE, "auto_center", "Centrado Automatico", True),
                        (PF_INT, "center_x", "Coor.Hor. del centro de la rotacion", 600),
                        (PF_INT, "center_y", "Coor.Vert. del centro de la rotacion", 600),
                        (PF_TOGGLE, "interpolate", "Interpolacion", True),
                        (PF_RADIO, "clip_result", "Resultado", "0", (("TRANSFORM-RESIZE-
ADJUST", "0"),("TRANSFORM-RESIZE-CLIP","1"),("TRANSFORM-RESIZE-CROP","2"),("TRANSFORM-
RESIZE-CROP-WITH-ASPECT","3")))
                ],
                [],
                rotar)
        main()
```

inclinar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite inclinar una imagen, y alamcena los datos de las acciones
efectuadas en un fichero .xml
# definimos las funciones necesarias
def inclinar(img, drawable, shear_type,magnitud,interpolate,clip_result):
        #ejecucion de la accion
        pdb.gimp_drawable_transform_shear_default(drawable,
shear_type,magnitud,interpolate,clip_result)#Ejecucion del procedimiento

        procedimiento="gimp_drawable_transform_shear_default"# se guardan en variables el
nombre del procedimiento y los parametros usados.

        parametro1="shear_type"
        parametro2="magnitud"
        parametro3="interpolate"
        parametro4="clip_result"
        v_parametro1=shear_type
        v_parametro2=magnitud
        v_parametro3=interpolate
        v_parametro4=clip_result

        #Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
        try:
                file = open("archivo.tmp.xml", "r")
        except:
```

```
                pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
        else:
                file.close()

                #Se abre el fichero y se añaden los datos
                f = open("archivo.tmp.xml", "a")

                f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
                f.write ("\t\t<parametros>\n")
                f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro2+"\" valor
=\""+str(v_parametro2)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro3+"\" valor
=\""+str(v_parametro3)+"\"/>\n")
                f.write ("\t\t\t<parametro name =\""+parametro4+"\" valor
=\""+str(v_parametro4)+"\"/>\n")
                f.write ("\t\t</parametros>\n")
                f.write ("\t</procedimiento>\n")

                f.close()

# función principal
if __name__ == '__main__':

        # llamada a función register
        register(
                "inclinar",
                "Inclinar la imagen",
                "Inclinar la imagen",
                "Jose Antonio",
                "Jose Antonio",
                "2011",
                "<Image>/Gestion de Tareas/Procedimientos/Inclinar",
                "RGB*, GRAY*",
                [
                        (PF_RADIO, "shear_type", "Tipo de inclinacion", "0",
(("ORIENTATION-HORIZONTAL", "0"),("ORIENTATION-VERTICAL","1"))),
                        (PF_FLOAT, "magnitud", "Magnitud de la inclinacion", 0.0),
                        (PF_TOGGLE, "interpolate", "Interpolacion", True),
                        (PF_RADIO, "clip_result", "Resultado", "0", (("TRANSFORM-RESIZE-
ADJUST", "0"),("TRANSFORM-RESIZE-CLIP","1"),("TRANSFORM-RESIZE-CROP","2"),("TRANSFORM-
RESIZE-CROP-WITH-ASPECT","3")))
                ],
                [],
                inclinar)
        main()
```

desaturar.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

# Script para Gimp en Python

# importamos los módulos necesarios
from gimpfu import *
import os

#Este plug-in permite convertir los colores de una imagen en niveles de gris, y alamcena
los datos de las acciones efectuadas en un fichero .xml
# definimos las funciones necesarias
def desaturar(img, drawable,modo):
        #ejecucion de la accion
        pdb.gimp_desaturate_full(drawable,modo)#Ejecucion del procedimiento desaturate

        procedimiento="gimp_desaturate_full"# se guardan en variables el nombre del
procedimiento y los parametros usados.

        parametro1="modo"
        v_parametro1=modo
```

```
	#Con este tratamiento de excepcion, se evita crear un nuevo fichero si es que no
existe anteriormente.Tambien se avisa al usuario de que aunque haga la accion sobre la
imagen , esta no se guardara.
	try:
		file = open("archivo.tmp.xml", "r")
	except:
		pdb.gimp_message("No se ha iniciado el proceso.No se guardara esta
accion.")
	else:
		file.close()

		#Se abre el fichero y se añaden los datos
		f = open("archivo.tmp.xml", "a")

		f.write ("\t<procedimiento nombre=\""+procedimiento+"\">\n")
		f.write ("\t\t<parametros>\n")
		f.write ("\t\t\t<parametro name =\""+parametro1+"\" valor
=\""+str(v_parametro1)+"\"/>\n")
		f.write ("\t\t</parametros>\n")
		f.write ("\t</procedimiento>\n")

		f.close()

# función principal
if __name__ == '__main__':

	# llamada a función register
	register(
		"desaturar",
		"Convertir los colores en niveles de gris",
		"Convertir los colores en niveles de gris",
		"Jose Antonio",
		"Jose Antonio",
		"2011",
		"<Image>/Gestion de Tareas/Procedimientos/Desaturar",
		"RGB*, GRAY*",
		[
			(PF_RADIO, "modo", "Seleccionar un poco de gris basado en : ",
"0", (("Claridad", "0"),("Luminosidad","1"),("Media","2")))
		],
		[],
		desaturar)
	main()
```