Desarrollo de Aplicación para Edición de Imagen Digital

DAVID GRAU, Universidad de Valencia

Se va a desarrollar una aplicación que pueda substituir la que se ha utilizado hasta ahora en prácticas de Tratamiento Digital de Imagen, PDI32. Este nuevo software está adaptado a sistemas operativos actuales como Windows Vista, 7 y 8 con una interfaz de usuario acorde con estos ellos. Puesto que el programa se utilizará en prácticas debe incluir como mínimo todas las operaciones y efectos que se utilizarán la asignatura: adquisición de imágenes, realzado por procesado por punto, filtrado espacial, filtros para segmentación, filtros morfológicos y diferenciación de objetos. Al ser una nueva versión del programa se va aprovechar para añadir nuevas utilidades como funciones de dibujo, procesado por lotes y transformada de Fourier. Todas estas funciones se incluyen en una sola clase que agrupa todos los métodos necesarios para realizar estas tareas. El proyecto final está escrito de forma que se pueda utilizar como manual de consulta para futuros alumnos incluyendo la teoría necesaria para realizar estas operaciones y el código fuente explicado.

Palabras clave: Procesado de imagen digital, procesado por punto, filtrado espacial, filtro de segmentación, diferenciación de objetos, transformada de Fourier.

1. INTRODUCCIÓN

En prácticas de Tratamiento Digital de Imagen se ha estado utilizando un software programado por la Universidad Politécnica de Valencia llamado PDI32. Este fue escrito para Windows XP y se ha comprobado que es inestable en sistemas operativos actuales como Windows Vista, 7 y 8. Además se ha observado que la aplicación tiene ciertas carencias a la hora de ser utilizado en estas prácticas ya que no contaba con todos los procesos y efectos que se requieren.

Por ello se ha propuesto el desarrollo de un software nuevo que pueda substituir a PDI32 ampliando sus funcionalidades, mejorando su interfaz de usuario, complementándolo con un extenso archivo de ayuda que explique con mayor profundidad todos los procesos que se pueden utilizar y que contenga ejemplos, también se contará con una librería de archivos de presets para ciertas utilidades y el propio proyecto se podrá utilizar para profundizar en la compresión de estos procesos.

El objetivo de la asignatura es programar un sistema de reconocimiento de formas simple así que este también se va a implementar en el proyecto para poder comprobar que el código escrito en las prácticas funciona correctamente.

Las especificaciones básicas del proyecto son:

— Mejora de la interfaz de usuario.
— Realzado de imágenes por procesado por punto.
— Filtrado espacial.
— Filtros para segmentación.
— Filtros morfológicos.
— Diferenciación de objetos.

Se ampliarán estas especificaciones iniciales con:

— Herramientas de dibujo y portapapeles.
— Captura de imagen.
— Transformada de Fourier.
— Procesado por lotes.

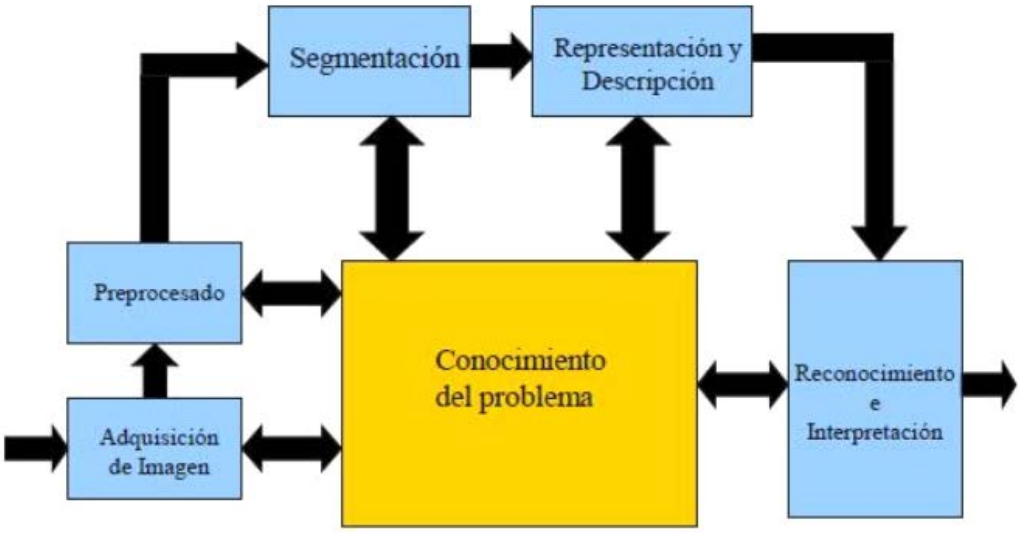

Figura 1

En la figura 1 se puede observar el proceso necesario para llegar a realizar el reconocimiento de formas. El problema de nuestro sistema es diferenciar si la forma de un objeto es alargada o redonda. A partir de este se aplicarán una serie de procesos que parten de la adquisición de la imagen ya sea capturando el escritorio o por medio de webcam y a partir de esta información realizar los procesos necesarios para que nuestro sistema pueda obtener los datos de perímetro y área con los que poder decidir la forma del objeto que estamos analizando.

Como ya sabemos [5] las imágenes digitales se guardan en matriz de bytes. En nuestro caso se van a utilizar profundidades de color de 8 bits en escala de grises y de 8 bits por canal para imagen en color a la hora de realizar los cálculos que requieren los procesos utilizando siempre el espacio de color RGB [5].

Una vez se ha obtenido la imagen se debe pre procesar mediante filtros de realce por procesado por punto para mejorar la calidad de esta. Esta parte del proceso de reconocimiento no es siempre la misma. Dependiendo de la imagen con la que estemos trabajando habrá que aplicar unos filtros u otros ya que uno puede ser muy útil para una imagen pero puede afectar gravemente a los datos de otra. Aquí se utilizará filtrado por procesado por punto [1] y filtrado espacial que trabaja con grupos de píxeles y está basado en Sistemas Lineales Invariantes en el Tiempo LIT [4]. Para tener mejores resultados será necesario apoyarse en el histograma ya que puede dar valores exactos sobre los que aplicar los efectos [1].

Parte del campo de la visión artificial es la segmentación de imágenes que nos permite dividir una imagen en varios objetos y la simplifica para facilitar su análisis [1]. Para ello se utilizarán los filtros de segmentación que buscarán pixeles que tengan características parecidas detectando discontinuidades. Con estos se reduce la información de la imagen de forma que vemos los límites de los objetos contenidos en ella con mayor claridad. Será posible detectar líneas o bordes de objetos.

El siguiente paso es pasar la imagen por filtros morfológicos. En matemáticas se utiliza la palabra morfología para designar una herramienta que se utiliza para

extraer los componentes de una imagen empleados para representar y describir la forma de su región [1].

Estos filtros se aplican sobre imágenes en blanco y negro y se utilizan operaciones booleanas para realizarlos. Cambian el tamaño de la imagen procesada con el objetivo de suavizar sus contornos eliminando ciertas estructuras como agujeros, bultos… El objetivo es simplificar la imagen preservando sus características esenciales de forma y eliminando irrelevancias.

Un ordenador tiene la ventaja de que puede superar las capacidades de visión humanas pero no existen sistemas de visión por ordenador totales, esto quiere decir que solo se pueden aplicar a fines concretos por lo que sí pueden resultar muy útiles para trabajos automáticos [1].

El sistema va a necesitar calcular el área y el perímetro del objeto. Esta tarea ya se puede hacer tras pasar por los filtros morfológicos. Para poder reconocer la forma del objeto se utilizará un descriptor conocido como densidad de una región o compacidad [1] que se puede calcular con la ecuación (1).

$$c = \frac{perímetro^{\,2}}{área} \tag{1}$$

Si el valor de C es 4π la forma del objeto será redonda, si el resultado es mayor el objeto tratado es alargado.

Una parte de las prácticas trata sobre reparación de imágenes distorsionadas o con errores ya sea debido a problemas en la adquisición o porque los datos se han corrompido. Para solucionar estos problemas se utiliza la Transformada de Fourier (TF) [4] así que también se ha añadido esta funcionalidad. Como no es parte de las especificaciones iniciales se ha optado por utilizar una librería externa que pueda darnos la información de la TF.

Como parte de las herramientas de dibujo se han incluido las posibilidades de cambiar el tamaño del lienzo, redimensionar la imagen y rotaciones. Puesto que es muy fácil cambiar el formato de las imágenes se ha optado por añadir una función de procesado por lotes que se encarga de recoger un grupo de imágenes y convertirlas a otro formato pudiendo elegir entre bmp, jpg, png y gif.

2. DESARROLLO

2.1 Entorno de Desarrollo

Para desarrollar la aplicación se ha optado por utilizar lenguaje C++ debido a su capacidad para trabajar con punteros. El software se va a utilizar sobre plataformas Windows así que el entorno debe compilar correctamente para ellas y a ser posible que facilite lo máximo posible el diseño visual del programa.

Un entorno que cumple con estos requisitos es Embarcadero Rad Studio XE [6] utilizando el compilador de C++. Para el apartado visual este entorno incorpora una extensa biblioteca llamada Visual Component Library VCL que incluye los componentes de menú tipo Ribbon [3], estilos de ventana de dialogo nuevos y un potente editor visual para diseñar las ventanas. Además aporta una extensa ayuda sobre el API de Windows. También ha sido necesario descargar el compilador de ayuda en formato CHM (Microsoft Compiled HTML Help) [9] y el software Adobe

Dreamweaver para maquetar y compilar los archivos de ayuda con que contará el programa.

El desarrollo se va a hacer por partes. Del tratamiento de imagen se va a encargar una clase llamada utilidades la cual se divide en dos partes. Una de ellas para el procesado de imagen en sí y otra que contiene las funciones de dibujo que están programadas de otra forma mediante el API de Windows.

2.2 Clases utilizadas

Se van a utilizar muchas de las clases que proporciona la VCL [2] para tratamiento de gráficos ya que así el programa será más robusto y más rápido, además de ahorrar trabajo.

— La clase TCanvas provee de una zona de dibujo abstracta que a su vez nos da herramientas de dibujo básicas como rectángulo, elipse… Y lo más importante un buen control sobre la paleta de colores.

— La clase TColor. Se van a utilizar 4 bytes por color pero dejando el primero a cero ya que no se utiliza el canal de transparencia. Los siguientes 3 bytes son los valores RGB aunque también es posible referenciar los colores por sus nombres.

— TPen se encarga de los atributos de las líneas pudiendo seleccionar color, grosor y estilo.

— TBrush es para el relleno de objetos pudiendo especificar su color y estilo.

— Sobre la capa de dibujo se va a poner otra capa de ayuda. Esta capa sirve para dibujar previsualizaciones de las herramientas de dibujo, recortes, tamaño y forma del pincel… Se utiliza esta debido a que su velocidad de refresco es mayor, incluye doble buffer y lo más importante, es transparente así que se puede superponer a la zona de dibujo.

— La zona de dibujo en sí utilizará la clase TImage que incluye TCanvas para dibujar y TPicture.

— TPicture es el contenedor de imágenes y se encarga de cargar y guardar imagen en diversos formatos.

— Para almacenar los mapas de bits se cuenta con TBitmap. Esta clase tiene toda la información de la imagen en sí como tamaño, profundidad de bits, paleta. También tiene las funciones de acceso a pixel que serán las encargadas de todo el trabajo así como el método asign que nos permite copiar TBitmaps.

2.3 Interfaz de usuario

Esta aplicación utilizará 17 ventanas para los distintos procesos que puede realizar. Para ahorrar código se ha creado la ventana diálogo que es configurable, figura 2. Al llamar a esta ventana se le pasarán los datos de configuración visuales y su título entre otros parámetros y un objeto de la clase Utilidades con la imagen y las funciones de dibujo y procesado de imagen.

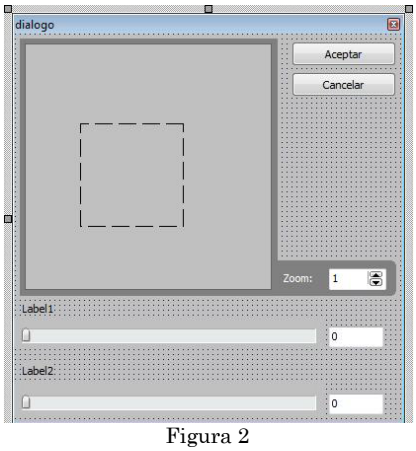

Figura 2

2.4 Putpixel y Getpixel

Son dos métodos críticos en todo programa de tratamiento de imagen ya que en todo momento es necesario tanto leer como escribir pixeles en las imágenes con las que se están trabajando. Para elegir como se haría este método se ha calculado experimentalmente la velocidad de las distintas posibilidades con las que se contaba. Para ello se ha escrito un pequeño programa que hace lecturas y escrituras sobre imágenes de diferente tamaño con una profundidad de color de 24 bits por pixel. Se han probado los métodos Pixels de la clase TCanvas, ScanLine de la clase TBitmap y GetDibBits del API de Windows Los resultados obtenidos se pueden ver en la tabla 1.

Tabla I. Medidas de tiempos (ms)

Tamaño	100x100	400x400	1600x100
Pixels	4700	81300	81300
ScanLine	31	296	218
GetDibBits	187	1539	1735

A la vista de los resultados se va a utilizar la función ScanLine ya que es, con diferencia, la más rápida. Además su funcionamiento también ayuda a mejorar la velocidad ya que devuelve líneas completas (scanlines) y el acceso a un array es muy rápido.
Sabiendo cómo utilizar este método se han construido tres funciones tanto para lectura como para escritura, una para escala de grises y otras dos para RGB. En RGB hay una que trabaja con los valores de color por separado y otra con la clase TRGBTriple que encapsula estos datos. Se utilizan estas dos opciones porque cada procesado necesita la información guardada de una forma u otra.

2.5 Luminosidad

Es un método muy simple pero muy utilizado. Algunos procesos requieren que se trabaje con el parámetro luminosidad de la imagen también es más rápido utilizar este valor único en lugar de procesar por separado los tres canales de color. Para ello se aplica la fórmula 2 obtenida de [1] a la que se le pasan los tres valores RGB del pixel a tratar.

$$luminosidad = (0.299 * R) + (0.587 * G) + (0.144 * B) \quad (2)$$

2.6 Histograma

Esta es una herramienta que se utilizará a menudo (figura 3). Tiene que trabajar tanto a nivel de brillo en imágenes en escala de grises también lo tiene que hacer en imágenes en color aplicando las misma operaciones sobre cada canal de color.

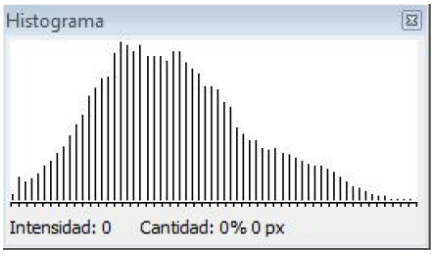

Figura 3

Al mover el cursor del ratón por encima se indica el valor de brillo, la cantidad de pixeles con este y su porcentaje en la imagen. También se ha escrito un método para que calcule y dibuje rápidamente la información ya que cada vez que se haga un cambio en la imagen se tiene que actualizar.

2.7 Procesado de Imagen por Punto

Este es el tipo de procesado más simple. Se recorrerá la imagen en sentido de lectura occidental y se aplicará un proceso al pixel seleccionado sin tener en cuenta los de alrededor. Son los efectos básicos para realzar la imagen trabajando sobre todo en los niveles de brillo.

Estos son: negativo, brillo, contraste, compresión de rango dinámico, umbralización, procesado del histograma y umbralización [1].

2.8 Filtrado espacial

Para este tipo de filtrado se utilizarán kernels estándar [1] que se han guardado previamente en matrices en la clase utilidades. Se recorre la imagen como en el procesado de imagen por punto pero en este caso se utilizará un pixel central y sus 8-vecinos [5] para aplicarles el kernel correspondiente por medio de la convolución 2D.
También es posible crear filtros definidos por el usuario con tamaños de kernel variables (figura 4). Esta ventana permite introducir los valores, utilizar valores aleatorios y calcular automáticamente el divisor del valor central del kernel.

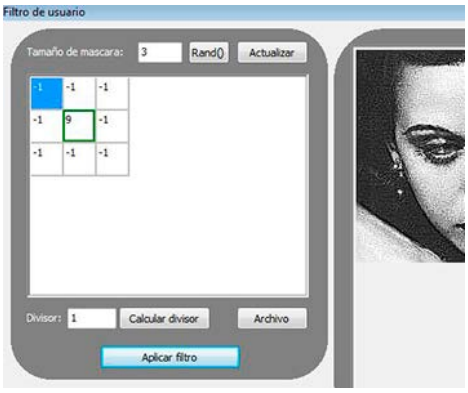

Figura 4

Los filtros principales serán el filtro paso bajo que realza las frecuencias bajas atenuando las altas y dando un efecto de suavizado a la imagen y el filtro paso alto que hace la operación contraria intensificando los detalles finos. Otro filtro que cabe en esta categoría es el de mediana que captura el pixel central y sus 8-vecinos, pasa esta información a un vector ordenado y se queda con el valor central de este de forma que se puede eliminar el ruido impulsivo, pixeles aleatorios muy separados entre sí.

2.9 Filtros de Segmentación

Este tipo de filtros trabajan con las mismas funciones que los filtros espaciales ya que también se aplican kernels pero se agrupan en otra categoría ya que no son parte del realzado de imagen, estos corresponden a la segmentación por detección de discontinuidades.

Aquí hay dos grupos. Filtros de detección de líneas que por medio de un kernel pueden dar como resultado solo las formas verticales u horizontales de la imagen y los filtros de detección de bordes que trabajan con gradientes. Estos aplican un kernel vertical y otro horizontal y pasan el resultado a una de las siguientes fórmulas.

$$Raíz = \sqrt{vertical^2 + horizontal^2} \tag{3}$$

$$Valor\ absoluto = |vertical| + |horizontal| \tag{4}$$

Estas fórmulas se aplican a las máscaras obtenidas por los kernels. La fórmula raíz proporciona un valor más exacto y valor absoluto es más rápida.

Otro filtro detector de bordes es el laplaciano, es el más simple de todos ya que solo hay que aplicar un kernel. El resultado es similar a la suma de los filtros de detección de líneas. Trabaja con la segunda derivada y da un resultado más exacto pero con el problema de que se añade a la imagen final.

2.10 Filtros morfológicos

Se utilizarán operaciones lógicas del algebra de boole concretamente las funciones AND, OR y XOR. También es importante la vecindad entre pixeles que rodean a uno central. Hay varios tipos de vecindad pero para estas operaciones solo se utilizará 4 8-vecinos, recogiendo los valores del pixel al que rodean.

Así tenemos cuatro filtros morfológicos. Erosión, que elimina penínsulas delgadas y objetos pequeños, al recorrer la imagen se pondrá a 1 el pixel central si sus vecinos están a 1. Dilatación elimina discontinuidades y huecos aumentando el tamaño de los objetos. Se pondrá a 1 el pixel central cuando al menos un vecino esté a 1. Al combinar estos filtros se obtienen los dos restantes que son Open que elimina puntos aleatorios (ruido negro) y salientes estrechos mediante erosión y luego dilatación y Close que suaviza la imagen rellenando salientes estrechos y eliminando el ruido blanco haciendo dilatación seguida de erosión.

2.11 Diferenciación de objetos

El proceso final de reconocimiento de formas requiere los siguientes pasos:
(1) Limpiar la imagen con los filtros necesarios.
(2) Pasar la imagen a blanco y negro con umbralización.
(3) Segmentar la imagen.
(4)El objeto debe estar dibujado en blanco sobre un fondo negro así que si es necesario se aplicará la función negativo a la imagen.
(5) Eliminar los posibles elementos que se confundan con el fondo mediante close.
(6) Eliminar salientes utilizando la operación open.

Una vez seguidos estos pasos se puede proceder al cálculo de la compacidad (fórmula 1). Para obtener el área de un objeto se contarán todos los pixeles blancos que lo componen. El perímetro se puede calcular de dos formas, por gradientes o con morfología. Por gradientes se sacará el gradiente horizontal y vertical combinándolas mediante valor absoluto (fórmula 4) y finalmente se umbraliza para que los pixeles del perímetro tengan 1 pixel de grosor. Con morfología primero se procede a erosionar el objeto y se termina aplicando la operación booleana XOR entre la imagen original y la erosionada. Una vez se han hecho estos procesos se pueden contar los pixeles blancos que quedan que representan el perímetro del objeto a analizar.

Finalmente se aplica la formula 1 con estos valores y se obtiene la compacidad.

2.12 Herramientas de dibujo

Las herramientas de dibujo básicas son pincel, dibujo de líneas, rectángulos y elipses, para ello se utilizan las clases TPen y los métodos que proporcionan TCanvas. La paleta de colores viene en la VCL así que solo hay que abrir la ventana de diálogo para acceder a ella, en el caso de imágenes en escala de grises se ha optado por utilizar una imagen que representa toda la escala de grises y capturando el pixel sobre el que está el ratón.

También se han implementado las funciones de recorte, copia y pegado que funciona con la clase ClipBoard y utiliza el método copyRect del API de Windows así como un sistema para deshacer acciones simple. Para ello se mantiene siempre una copia de la imagen antes de ser procesada de forma que se puede pegar sobre la obtenida y volviendo hacia atrás, si se vuelve a repetir la operación se vuelve a la imagen procesada.

Hay un conjunto de funciones que se han incluido como herramientas de dibujo ya que no entran en ninguno de los grupos anteriores y están mas orientadas a estas tareas como cambiar el tamaño del lienzo, cambiar el tamaño de la imagen (con dos modos, repitiendo pixeles o con interpolación lineal), volteado de imagen, rotaciones fijas y libres.

Para la interpolación lineal el proceso que se ha seguido es ampliar el tamaño de la imagen de forma que los pixeles quedan separados por espacios en blanco y se utiliza la fórmula 5 para calcular el valor de los valores intermedios.

$$incremento = \frac{colorFin - colorIni}{Escala} \tag{5}$$

2.13 Captura de Imagen

El primer paso del proceso de reconocimiento de formas es la captura de imagen. Es posible capturar la pantalla del ordenador o por medio de webcam. Para capturar la pantalla se ha utilizado el método del API BitBlt pasándole como parámetro de captura el valor 0 que es el que referencia a la pantalla. La captura por webcam se ha desarrollado utilizando la librería Video For Windows que facilita todas las funciones de control multimedia necesarias. Se han utilizado básicamente dos funcionalidades de esta. Abrir la ventana de configuración de captura desde la que se puede seleccionar la webcam y ajustar parámetros de esta como brillo o contraste y los controles de cámara. Para poder utilizarla hay que asignarle una superficie de dibujo, luego se enciende la cámara y comienza a dibujar en esta, para la captura se recurre a la función bitBlt y cuando se finaliza el proceso se desconecta.

2.14 Transformada de Fourier

Puesto que esta función no era un requisito básico de la aplicación se ha optado por utilizar una librería de terceros llamada FFTW [8] que es libre.

Para poder utilizarla se tiene que pasar toda la información de la imagen a un vector. Una vez ha calculado la FFT devuelve el resultado en un array de dos dimensiones cuya primera fila contiene los valores reales y la segunda los imaginarios. Para hacer la representación gráfica se calculará el módulo y el logaritmo. El resultado de estas operaciones se tiene que normalizar a valores de 0 a 255 para poder utilizar paleta de color indexado. Una vez normalizados los valores se tiene la parte superior de la FFT así que se puede dibujar directamente sobre u lienzo para posteriormente dibujar la misma información invertida y obtener el gráfico completo.

Los valores que se han calculado se pueden exportar a formato HTML o en CSV para poder importarlo posteriormente a Excel.

3. CONCLUSIONES

Al finalizar el proyecto se ha comprobado que se han superado los objetivos principales ya que se han añadido muchas funcionalidades nuevas. Con el tiempo se han encontrado errores que no afectan al funcionamiento normal ni a la estabilidad del programa en sí.

Puesto que se ha podido diseñar el archivo de ayuda mediante HTML se ha obtenido un presentación muy cuidada y se han podido insertar con facilidad todos los ejemplos de uso y el material requerido para finalizar las prácticas de la asignatura.

Aunque se han conseguido todos los objetivos ya se han propuesto líneas futuras para aprovechar el trabajo realizado siendo el principal objetivo aumentar la velocidad de procesado utilizando un acceso a pixeles más rápido. También se ha pensado en utilizar formatos de color de 32bits para poder aprovechar el canal alfa y poder trabajar con transparencias y por lo tanto con capas.

REFERENCIAS

[1] Ignacio Bosch Roig, Pablo Sanchis Kilders, Jorge Gosálbez Castillo, José Javier López Monfort. Prácticas de tratamiento digital de la imagen. Valencia. Editorial UPV. 2009.
[2] Marco Cantú. Delphi 2009 Handbook. Wintech Italia. 2009.
[3] Marco Cantú. 2008. Building User Interfaces with Delphi. Wintech Italia. 2009.
[4] Foley, Van Dam, Feiner, Hughes. Computer Graphics: Principles and Practice. Addison-Wesley Publishing Company. 1990.
[5] William K. Pratt. Digital Image Processing. Wiley. 2007.
[6] RAD Studio XE documentation Wiki. Disponible en: http://docwiki.embarcadero.com.
[7] Win 32 API. Disponible en: http://msdn.microsoft.com/es-es/library/windows/desktop/ff818516(v=vs.85).aspx.
[8] FFTW library documentation. Disponible en: http://www.fftw.org
[9] Microsoft HTML Help. Disponible en: http://msdn.microsoft.com/en-us/library/windows/desktop/ms670169(v=vs.85).aspx.