ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

TÉCNICAS DE SUPERRESOLUCIÓN DE IMÁGENES Y SU IMPLEMENTACIÓN EN ANDROID

Miguel Fuertes Fernández

Miguel Pagola Barrio

Pamplona, 30 de Agosto de 2011

ÍNDICE

INTRODUCCION

CUANDO LOS MÓVILES SE CONVIRTIERON EN SMARTPHONES.

Se dice que uno de los primeros teléfonos móviles, fue el Handie Talkie H12-16 de Motorola, que funcionaba a una frecuencia de 60mhz. Mucho ha llovido desde entonces.

Cuando realmente empezamos a hablar de la telefonía móvil, tal y como la conocemos ahora, es a partir de los años 90, cuando aparecieron los primeros estándares de telefonía móvil, que ya trabajaban a unas frecuencias de entre 900mhz y 1800mhz, permitiendo una mayor calidad de audio al mismo tiempo que un menor tamaño en los terminales que lo soportaban. Esta red de telefonía, que también soportaba datos (aunque a un ratio muy bajo), es la que se conoce con el nombre de 2G.

Fue aquí, cuando salieron los primeros terminales, portátiles, con pantalla de liquido y sonido monofónico, como son los conocidos Nokia (3310, 3210) o la familia de Ericsson como el t10.

Tras ese boom inicial, que supuso un avance en la comunicación, la demandas de servicios telefónicos creció. Ya no bastaba con voz y SMS. Así nace lo que se conoce popularmente como 2.5G.

El “no” estándar 2.5G, es una forma de llamar a los terminales que ofrecían servicios a caballo entre el estándar 2G (GSM) y el futuro estándar 3G (UMTS, WDCMA). Así, esa demanda de servicios, permitió la creación de nuevos estándares como el MMS y EMS, necesitando tecnologías como el GPRS y el EDGE.

Estas nuevas tecnologías, están orientadas al trafico de datos por la red, permitiendo velocidades de 56kbs a 114kbs en el caso del GPRS y de hasta 384kbs en el caso del EDGE. Pero lo interesante de la tecnología, no es solo la tecnología en si, sino lo que te permite hacer.

Esta fue la época en la que empezamos a ver los primeros terminales con cámaras de fotos, ya que las redes lo soportaban, y los estándares estaban creados, me refiero, por supuesto al estándar MMS.

MMS, o Multimedia Messaging System, es el estándar que lanzo la implantación de las cámaras en las telefonía móvil. La idea inicial surge con la intención de compartir no solo texto o voz, sino cualquier contenido multimedia.

Aquí nacen móviles como el J-SH04 de Sharp Corporation, primer teléfono con cámara, el Nokia 7650 con una resolución de 0.3MPx o el mas conocido Nokia n70 con una resolución de 2MPx.

Para la época, la idea de tener cámara en el móvil, era a la vez absurda e innovadora. Era la época en que empezaban a existir cámaras digitales, la calidad de las imágenes digitales era bastante pobre, y por eso las compañías tuvieron que hacer un gran esfuerzo para impulsar esta tecnología.

Eslóganes como "comparte el momento" se habrían podido aplicar a este contexto, puesto que cuando el GPRS, el MMS y las cámaras en los móviles empezaron a existir, lo hicieron de la mano.

Recapitulando, nos encontramos en el nuevo siglo, y tenemos teléfonos capaces de tomar fotos, tenemos redes de datos que soportan el envío de dichas fotos con estándares, y tenemos redes de comunicación que soportan voz, con bastante buena calidad. ¿Cuál es el siguiente paso?

El siguiente paso, es no tanto para las operadoras y las redes móviles, sino para los terminales en si. Si bien es cierto que nace el estándar 3G, utilizando UMTS sobre redes WDCMA con una tasa de transferencia de 114kbps a 7.2mbps, este hecho no es muy importante porque la demanda no esta en estos parajes.

Un "daño" colateral de estos nuevos estándares y estas nuevas velocidades, hizo que los terminales se actualizasen para soportar este nuevo tipo de redes. En esta época la tecnología ha avanzado bastante, y ya podemos encontrar cámaras digitales con muy buena calidad. Este avance en la tecnología nos ha llevado a los Smartphone. Teléfonos inteligentes que (casi) dejan de ser teléfonos. Ha cambiado radicalmente el concepto de telefonía móvil. Ya no es un aparato que sirve para comunicarse, es mucho mas.

Actualmente no se ha superado la red 3G. Se habla del uso de HDSPA como internet móvil, pero no ha salido ningún estándar 4G, aunque es esta probando con el llamado LTE. La demanda ha cambiado. Ya no se busca voz y texto solamente, ahora se busca internet, por eso la demanda de teléfonos ha cambiado.

LOS SMARTPHONE: IPHONE, WINDOWS PHONE Y ANDROID.

Como hemos comentado en la evolución histórica, la telefónica y el concepto que tenemos de terminales telefónicos ha cambiado desde su origen hasta nuestros días. Hoy día podemos encontrarnos con terminales que compiten en versatilidad con los ordenadores personales. ¿Por qué ha pasado esto? Gracias a una serie de hitos importantes, como son: Windows Mobile, Palm, Blackberry, y posteriormente IOS y Android.

SMARTPHONES: MÓVILES CON SISTEMA OPERATIVO.

Llevamos un rato hablando de los smartphones, pero no los hemos definido. Sabemos que nacen de la necesidad de tener internet en el móvil. Sabemos que no son solo voz y datos sino que son mucho mas, pero no hemos dado una definición clara de lo que son. Son todo esto y mas. De las muchas definiciones que existen, a mi me gusta definirlos así. Los Smartphone son ordenadores. La gran diferencia que tienen con los móviles, aparte del procesador, la memoria RAM, etc. Es el sistema operativo de que estos disponen, aumentando su versatilidad.

Ahora que hemos llegado a los sistemas operativos, revisemos un poco la historia de estos en los teléfonos inteligentes, porque es interesante para centrar este documento en un contexto.

Nokia, quien tenia gran experiencia con móviles, se lanzo al terreno de los teléfonos inteligentes, con sus primeros teléfonos con sistema operativo SymbianOS. Aunque aun estábamos en la segunda generación de teléfonos (n70) la cosa prometía, puesto que ya se podían instalar aplicaciones. Las comunicaciones de aquellos aparatos distan mucho de las de ahora, la conectividad era o gsm o bluetooth. Symbian ha permanecido siempre en la sombra como teléfonos de gama alta, pero no se les ha considerado realmente smartphones hasta ahora con el relanzamiento de su sistema operativo.

En el año 2000 Microsoft saco al mercado su sistema operativo para teléfonos llamado Pocket PC y después conocido como Windows Mobile, como idea de llevar al mercado de los móviles su sistema operativo implantado en los ordenadores.

Esto ya era un hito, puesto que cambiaba el significado de los móviles. Hasta la fecha, el fabricante creaba un software que se implantaba en el teléfono (firmware) que rara vez se actualizaba y que tenia las aplicaciones necesarias para el correcto funcionamiento del teléfono. Con Windows Mobile se abrió un mundo de posibilidades, por la posibilidad de instalar aplicaciones en el teléfono, y por lo tanto de programarlas. El teléfono adquiría un nuevo significado, puesto que no solo hacia lo que el fabricante quería, sino lo que el usuario deseaba. Palm, la competencia, también saco su sistema operativo ya implantado en las agendas electrónicas.

Lo que Windows Mobile suponía a nivel de usuario, BlackBerry supuso a nivel ejecutivo/empresarial. Si bien no supuso el boom de Windows Mobile, en la actualidad, al reencauzarse a otro mercado, esta resurgiendo como proveedor de smartphones.

Los servicios ofrecidos son los mismos. Capacidad de instalación de aplicaciones, funciones de agenda, acceso a internet, etc.

Pero cuando Windows creía que tenia la gallina de los huevos de oro, le salieron dos serios competidores. Primero apareció su archienemigo, Apple, y creo otro hito en esta historia. Con su iPhone, demostró que el mundo de los smartphones no era cosa de entendidos, que todo el mundo podía tener un ordenador en la palma de su mano.

Era un terminal, con mucha potencia y con gran facilidad de uso. Así mismo, Apple puso mucho interés en que la comunidad de desarrolladores trabajase a fondo, y pronto se pudieron ver gran cantidad de aplicaciones. Su facilidad de uso, sus aplicaciones, y sus múltiples opciones en el campo de las telecomunicaciones hicieron de él el buque insignia de Apple, y lo lanzaron en el mercado.

Aunque Apple ha sacado al mercado versiones actualizadas del aparato, el sistema operativo (aunque actualizado) siempre ha sido IOS. El hecho de que el aparato es siempre conocido, y de que el sistema operativo solo se usa en dichos aparatos, ha conseguido que los desarrolladores crearan software especifico para los aparatos sacando el máximo beneficio de los mismos.

En la actualidad Apple ha sacado 5 iPhones, siendo el iPhone4 el ultimo de ellos. Además se esta abriendo al mercado creciente (y en mi opinión iniciado por ellos) de las tabletas, con sus iPads, teléfonos inteligentes mas grandes, pero sin teléfono.

Windows Mobile ya estaba casi olvidado, y Apple ya había hecho su jugada, cuando entro a competir otro sistema operativo, esta vez del gigante web Google, quien en 2005 saco al mercado su sistema operativo Android.

Si hemos dicho que Windows Mobile era Windows en los smartphones y IOS era MAC en los smartphones, Android en este caso es Linux en un Smartphone, con la potencia que eso supone.

Si bien IOS solo podía ser implantado en los teléfonos de la compañía de Cupertino, Android como Windows Mobile, podía ser implantado en varios teléfonos según fabricantes.

Actualmente Windows ha hecho una contra jugada sacando su Windows Phone 7 así como Nokia relanzando SymbianOS con su Symbian Anna. Solo el futuro sabe quien ganara en esta lucha.

ANDROID: DESDE EL LADO DEL DESARROLLADOR.

Puesto que la plataforma de desarrollo del proyecto es Android, centrémonos en este sistema operativo. Como bien he dicho es bastante reciente, fue creado y es

mantenido por Google, y actualmente se encuentra en la versión 2.3 GingerBread para smartphones y en la 3.1 HonneyComb para Tablets.

Como ya hemos adelantado es un sistema operativo basado en Linux, sin embargo, el desarrollo y la mayoría de aplicaciones que tiene, se ejecutan sobre una maquina virtual muy parecida a la de Java.

Google en internet te ofrece dos modalidades de desarrollo el SDK y el NDK. El primero, que es el que he utilizado, es el software development kit y es el que se basa en la maquina virtual (dalvik). Como veremos, la implementación de software en esta maquina virtual es muy parecido a Java (de hecho la sintaxis es la misma), con lo que facilita la portabilidad de aplicaciones de sobremesa y el desarrollo de programadores nuevos en Android.
Aunque se trate de una maquina virtual, la optimización de memoria esta muy trabajada, y la posibilidad de acceder a todos los recursos del sistema mediante objetos, es un punto muy favorable.

Por otro lado el NDK responde a native development kit. En este caso programamos aplicaciones para el sistema operativo, aplicaciones nativas que se ejecutaran fuera de la maquina virtual, sobre el Linux subyacente de Android. Esta opción es muy usada para la personalización de los terminales por parte de los proveedores de smartphones.

CÁMARA DE FOTOS EN TELÉFONOS INTELIGENTES.

Hemos dicho que la incursión de las cámaras de fotos en los teléfonos móviles se produjo cuando hablábamos del 2.5G. En aquel entonces las cámaras no eran el futuro, puesto que la calidad de imagen no era la que se esperaba de una cámara. Lamentablemente esto no ha cambiado.

En la actualidad la única forma de conseguir una cámara de fotos decente en un móvil, es adquiriendo uno de gama alta. El ciudadano medio, que se compra móviles de gama media, lo único que consigue son cámaras de fotos con resoluciones mediocres, estando la media entre 3.2MPx y 5MPx. No se malinterpreten las palabras, estas resoluciones, sobre todo la de 5MPx son bastante altas, sin embargo, en comparación con las cámaras tradicionales, la diferencia de calidad es abismal ganando estas ultimas la batalla.

Otro de los problemas, con los que nos topamos, es la mala óptica. Al ser un aparato que no solo es cámara de fotos, prima el aprovechamiento del espacio ante la calidad de la imagen, por tanto, la calidad de las lentes y la capacidad de aumento de las mismas deja mucho que desear. Si por ejemplo queremos hacer zoom sobre una parte del panorama, al no haber espacio el zoom que se hace es zoom digital, con técnicas de ampliación digital de la imagen. Al aumentar digitalmente el sensor, el ruido aumenta.

Acabamos de mencionar el ruido. Otro de los problemas importantes. A diferencia de la fotografía tradicional, en la que lo que se modificaba con la luz era analógico, y por tanto cogía toda la gama de colores , en el caso de las cámaras digitales, la imagen, la realidad, algo analógico, tiene que ser convertido en digital, discretizando los valores de los colores, y por tanto perdiendo calidad.

Estos sensores no se jactan de ser infalibles, con lo que en la mayoría de casos, las diferencias de temperaturas de color en el panorama, provocan en la imagen y en el sensor, cambios de funcionamiento creando ruido en la imagen.

TÉCNICAS DE SUPER-RESOLUCIÓN ANTE EL PROBLEMA DE LAS CÁMARAS.

Hemos visto que las cámaras de fotos en los móviles no son lo que el usuario espera de una cámara, vista en el concepto tradicional, por la cantidad de problemas que presentan.

En este proyecto lo que planteo es las técnicas de super-resolución de imágenes para paliar sobre todo el problema de las bajas resoluciones. De este modo el usuario tomaría imágenes con su no muy potente cámara de fotos y tras ejecutar el algoritmo correspondiente obtendría imágenes mas grandes con mas o menos perdida de calidad.

En este caso planteo dos algoritmos para la realización de la super-resolución, como son el algoritmo intervalar y la interpolación bicúbica, que mas adelante explicare.

OBJETIVOS DEL PROYECTO FIN DE CARRERA.

Por tanto, y tras esta introducción al proyecto, podemos decir que los objetivos del mismo son:

- Comprensión y aprendizaje de la programación en sistemas Android, en su maquina virtual. Acceso a la cámara y manejo de la misma y tratamiento de imágenes en esta plataforma.
- Comprensión e implementación de algoritmos de super-resolución como son la interpolación bicúbica y el algoritmo intervalar, para el aumento de resolución de las imágenes tomadas en Android.

ALGORITMOS DE SUPERRESOLUCION

Para intentar paliar el problema de la mala resolución en las cámaras fotográficas, podemos usar, como ya hemos dicho, técnicas o algoritmos de super-resolucion, que amplíen las imágenes, perdiendo no demasiada calidad. En este trabajo, expongo dos de estos algoritmos.

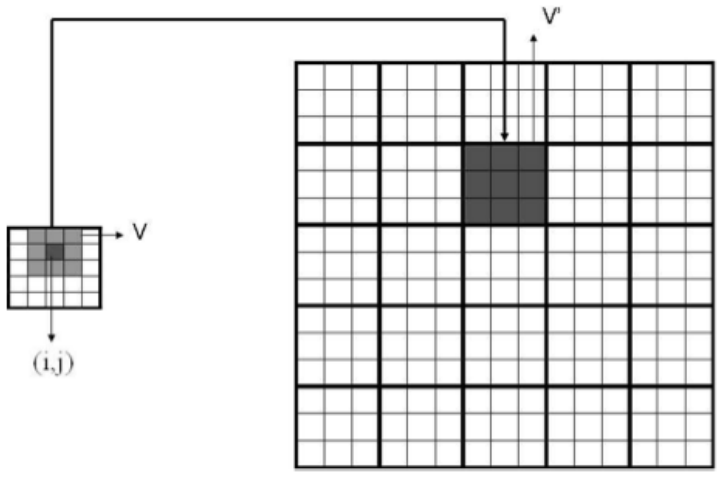
Por hacer un poco introducción a las imágenes en el ordenador, una imagen esta compuesta por pixeles, del ingles Picture-Elements, elementos de la imagen. En una imagen a color, en el formato mas común (RGB) un pixel se traduce en un valor en Hexadecimal (o decimal) que representa la intensidad de color en cada una de las tres capas de colores, a saber Rojo, Verde y Azul. Por tanto una imagen podemos representarla como una matriz (realmente tres) de intensidades de color, una matriz de enteros. Estas intensidades en decimal tienen valores entre 255 y cero, para una imagen en la que cada pixel ocupe 8 bits, representando 255 para blanco o toda la intensidad posible y 0 para negro o ninguna intensidad.

Siguiendo con las definiciones, expondremos también, ahora que sabemos que las imágenes son matrices, lo que se denomina como matriz normalizada. Una matriz o imagen normalizada es la que resulta de dividir todas las intensidades entre el valor de la máxima intensidad posible, que repito, para una imagen de 8bits de color es 255, obteniendo así valores entre 0 y 1, donde 0 vuelve a ser nada y 1 todo.

En este documento hablare de imágenes y matrices indistintamente.

ALGORITMO INTERVALAR.

Este primer algoritmo, es una aproximación bastante rápida a la solución deseada. La idea es que de cada pixel, cojamos una fila y una columna por debajo y por encima, de forma que seleccionemos una ventana de 3x3 pixeles.
Después, en la imagen solución, la ampliada, insertaremos esas ventanas transformadas a modo de “macropixeles” obteniendo así la proporción uno a tres y ampliando la imagen original.



Aunque en este caso hemos usado en todo momento ventanas de 3x3 podríamos usar ventanas mas grandes, siempre y cuando se pueda centrar el pixel actual (es decir ventanas cuadradas impares).

La idea seria ahora recorrer para cada matriz de intensidades de la imagen RGB, Todos los valores, seleccionar la ventana 3x3 que deje en el centro el pixel actual y aplicar una serie de operaciones matemáticas para aproximar los nuevos pixeles a su intensidad "real".

Vayamos paso por paso, supongamos (parte de) la imagen, representada por estos valores:

0.4	0.44	0.5	0.7	1
0.22	0.45	0.42	0.67	0.99
0.33	0.48	0.55	0.88	0.90
0.322	0.22	0.56	0.98	0.78

Suponemos que tenemos dos índices para recorrer las filas y las columnas (i: filas, j: columnas), y que hemos normalizado la matriz inicial, realicemos el algoritmo paso a paso para el elemento $\mathbf{t_{ij} = t_{23}}$:

1. Primero seleccionamos la ventana 3x3 correspondiente, dejando el elemento en el centro:

0.4	0.44	0.5	0.7	1
0.22	0.45	0.42	0.67	0.99
0.33	0.48	0.55	0.88	0.90
0.322	0.22	0.56	0.98	0.78

   (En el caso de estar en las zonas donde no podemos seleccionar toda la matriz, es decir en las esquinas y ultimas y primeras filas, en los valores de la ventana donde no tenemos valor, ponemos el valor del pixel central.)

2. Calculamos la amplitud máxima de los valores de la ventana, esto es el rango de valores con los que trabajamos, haciendo la diferencia del valor máximo de la ventana menos el valor mínimo:

   W = max(0.44 , 0.5 , 0.7 , 0.45 , 0.42 , 0.67 , 0.48 , 0.55 , 0.88) – min(0.44 , 0.5 , 0.7 , 0.45 , 0.42 , 0.67 , 0.48 , 0.55 , 0.88) = 0.88 – 0.42 = 0.46

3. Una vez que tenemos la amplitud de la ventana la multiplicamos por la desviación típica de toda la imagen, en el ejemplo tomaremos L = 0.87 por tanto **W = L*W = 0,87 * 0.46 = 0.40**.

4. Por ultimo, para terminar de formar la nueva ventana expandimos el pixel central combinándolo con los valores correspondientes de la ventana inicial: $\mathbf{t'_{ij} = central + t_{ij}*W}$ lo que por ejemplo, para el primer elemento significa: $\mathbf{t'_{11} = 0.42 + 0.44*0.40 = 0.596}$

0.44	0.5	0.7
0.45	0.42	0.67
0.48	0.55	0.88

   En el ejemplo, aplicando la formula, los nuevos valores serian los siguientes:

0.596	0.62	0.7
0.6	0.588	0.688
0.612	0.64	0.772

5. Insertamos el bloque entero en la nueva imagen, en la matriz ampliada, y seleccionamos el siguiente pixel de la matriz imagen inicial y repetimos el proceso.

Por tanto y resumiendo, el algoritmo es el que sigue:

```
Calcular la desviación típica de la matriz actual. (L)
PARA cada pixel HACER
        Seleccionar la nueva ventana.
        Calcular W como la resta entre el valor máximo y el mínimo
        Ajustar W con el valor de la desviación típica (L).
        PARA cada pixel en la ventana HACER
                (Pixel nueva ventana) := pixel + W*(pixel antigua ventana)
        FPARA
        Insertar la nueva ventana en la imagen resultante.
FPARA
```

INTERPOLACIÓN BICÚBICA

El segundo de los algoritmos aquí explicados es lo que se conoce como interpolación bicúbica. La idea de este algoritmo, o esta forma de interpolar pixeles, es muy sencilla. A partir de 16 puntos, generaremos una ecuación de tercer grado, una función, a la que luego pediremos los puntos extra que necesitamos.

Como en el caso del algoritmo anterior, tomaremos también ventanas de pixeles y a partir de estas ventanas de 16 pixeles interpolaremos los nuevos pixeles.
La interpolación cúbica de la que deriva la bicúbica es una "operación" matemática que extrapola la sencilla idea de dados dos puntos, sacar una recta a funciones curvas de 3 grado.

Expliquemos un poco el proceso de interpolación cúbica:

Conocemos el valor de la función y el de su derivada, en dos puntos de la función (x=1 y x=0), con lo que podemos plantear y sustituir las ecuaciones de 3 grado y su derivada:

$f(x) = ax^3 + bx^2 + cx + d$ $f'(x) = 3ax^2 + 2bx + c$	$f(0) = d$ $f'(0) = c$	$f(1) = a + b + c + d$ $f'(1) = 3a + 2b + c$

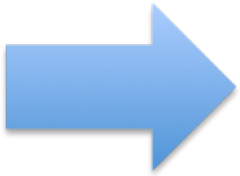
Podemos reescribir las ecuaciones de arriba en otras mediante operaciones matemáticas, despejando así los cuatro datos (a, b, c, d):

$a = 2f(0) - 2f(1) + f'(0) + f'(1)$
$b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$
$c = f'(0)$
$d = f(0)$

Aquí ya tendríamos las formulas para sacar la función y conseguir los nuevos valores. Si lo que estamos haciendo es para una lista de datos, con elegir 0 como derivada tendríamos suficiente, pero siempre es mejor hacer que el salto entre datos sea una línea curva, el resultado es mas ajustado.

Suponemos entonces que tenemos p0, p1, p2 y p3 en los puntos x=-1, x=0, x=1, x=2, podríamos usar las formulas de antes para interpolar los valores de f(0), f(1), f´(0) y f´(1), para interpolar puntos entre p1 y p2, y combinando con las cuatro anteriores sacamos los valores de los cuatro pesos:

$$\begin{aligned} f(0) &= p_1 \\ f(1) &= p_2 \\ f'(0) &= \frac{p_2 - p_0}{2} \\ f'(1) &= \frac{p_3 - p_1}{2} \end{aligned}$$



$$\begin{aligned} a &= -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3 \\ b &= p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3 \\ c &= -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2 \\ d &= p_1 \end{aligned}$$

Con esto habríamos explicado la interpolación cúbica lineal, es decir, para una ristra de datos, conseguir "su" función y generar mas datos. Para el escalado de imágenes lo que nos interesa es su forma matricial o interpolación bicúbica, que viene a ser lo mismo pero en dos dimensiones. Una vez entendida la interpolación cubica, la bicúbica no tiene mas complejidad que repetir los datos cuatro veces, como podremos ver en el ejemplo.

Antes de ver el ejemplo, presentemos el algoritmo:

```
Calcular el ratio de ampliación
PARA cada pixel en la nueva matriz HACER
        Calcular la posición entera respecto de la imagen antigua.
        Calcular los decimales de volver de la nueva posición a la antigua.
```

Esta diferencia son los nuevos puntos.
Seleccionar la nueva ventana. Desde el pixel de la antigua matriz actual.
Aplicar la interpolación cubica por filas, y guardar el resultado.
Usar como nuevo punto los decimales con respecto al ratio x
Aplicar la interpolación cubica con los resultados
Usar como nuevo punto los decimales con respecto al ratio y
Escribir el valor en la nueva matriz.
FPARA

Seleccionamos una matriz cualquiera, y en este caso no necesitamos que la matriz este normalizada:

112	223	33	4	111
33	**222**	44	31	65
166	12	55	211	123
122	123	44	244	245
145	135	167	189	199

A diferencia del algoritmo anterior, en el que la ampliación seria con ratio impar (3x, 5x, etc.) aquí nos seria indiferente. Para el ejemplo voy a volver a generar una matriz 3 veces mayor:

1. Calculamos el ratio de ampliación:
   5/15 = para las "x" y para las "y". (Lo necesitamos inverso por comodidad a la hora de operar)
2. Como estamos recorriendo la nueva matriz, tendremos que adaptar los índices de esta nueva matriz para coger las intensidades de la antigua matriz:
   Ant_indices = nuevos_indices * ratio (Mediante operaciones matemáticas nos hacemos también con la parte decimal de esta operación.)

   Tenemos entonces en antx y anty los índices de la antigua matriz, y en difx y dify los decimales.
   (En el ejemplo difx = 0.33 = dify)

3. Suponemos que tras la conversión de índices nos encontramos en el pixel con intensidad 222 de la antigua matriz, realicemos la interpolación bicúbica:

   Seleccionamos los valores de la fila anterior a las dos siguientes y ejecutamos la interpolación cúbica de dichos valores con difx:

   P[0] = cubica(112, 223, 33, 4, 0.33)= 176
   P[1] = cubica(33, 222, 44, 31, 0.33)= 184
   P[2] = cubica(166, 12, 55, 211, 0.33)= 7
   P[3] = cubica(122, 123, 44, 244, 0.33)= 92

Por ultimo hacemos la interpolación cúbica con los resultados y dify:

nuevoPixel = cubica(p[0], p[1], p[2], p[3], 0.33) = 129

4. Repetimos el proceso hasta completar la matriz.

Por ultimo y faltaría definir la función cubica() que he usado a fin de dar un ejemplo mas conceptual sin recargar con formulas. La función cúbica se encarga de generar los cuatro pesos a, b, c, d de las formulas anteriores, aplicarlos a los coeficientes y sustituir el valor ultimo en la ecuación inicial. Podríamos decir que su propio algoritmo seria:

1. Generar los pesos
2. Componer la ecuación: $f(x) = ax^3 + bx^2 + cx + d$
3. Devolver el valor f(a) donde a es el ultimo parámetro de la llamada, en el ejemplo anterior 0,33.

Una implementación en java de cubica():

```
public int bicubic (int[] p, float x) {
            return (int) (p[1] + 0.5 * x*(p[2] - p[0] + x*(2.0*p[0] -
5.0*p[1] + 4.0*p[2] - p[3] + x*(3.0*(p[1] - p[2]) + p[3] - p[0]))));
      }
```

MÍNIMA CONCLUSIÓN

Hemos explicado dos algoritmos de super-resolucion de imágenes. Pese a que el segundo tiene mas potencia, las imágenes se ven mucho mejor, el coste en memoria y ejecución es mayor que el primero, y por tanto es mas lento. Habría que hallar un compromiso entre calidad y tiempo.

IMPLEMENTACION

Hemos hecho una breve introducción histórica planteando los problemas actuales, y hemos descrito dos métodos de super-resolucion como solución a estos problemas. Veamos ahora como se traduce la teoría en la practica. En esta sección expondré el código y resaltare algunas ideas que me parece interesantes a la hora de programar en Android, así como pantallazos de la aplicación además de explicar como funciona.

Hemos comentado que cuando programamos sobre el SDK de Android las aplicaciones resultantes se ejecutan sobre una maquina virtual muy parecida a la de java. Google ha sido listo, y en vez de crear su propio entorno de desarrollo para Android, lo que ha hecho ha sido crear un plug-in para uno de los IDEs mas usados de java, como es Eclipse.

Como vemos que deriva de java y su sintaxis es similar, podemos decir que la programación es orientada a objetos, por lo que empezare por exponer las clases que conforman la aplicación Android.

Desde el punto de vista del desarrollador las aplicaciones Android tienen que tener una clase que extienda de la clase Activity. Obligándonos a implementar los métodos **oncreate, onresume, onpause**. Estos métodos son llamados cuando el usuario abre la aplicación, la pone en pausa o la cierra.

En los anexos podrá encontrar mainActivity.java. Esta es mi actividad, la clase principal. Voy a explicarla un poco:

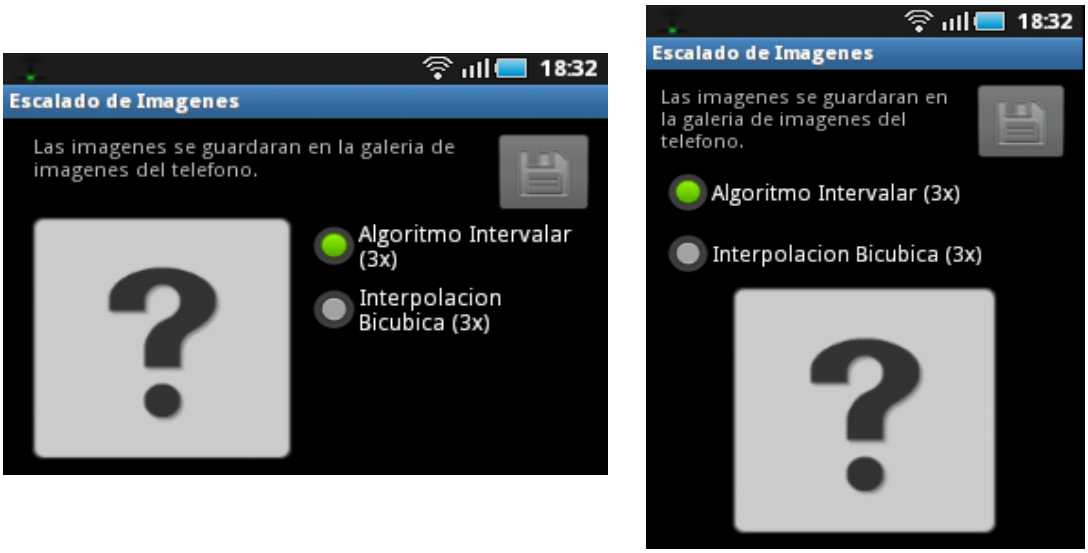
- Como hemos dicho extiende de Activity, pero también implementa tres interfaces mas, como son OnClickListener, para gestionar los eventos de click (Tocar la pantalla), Runnable, puesto que en un momento dado necesito que sea un hilo, y Handler.CallBack, porque en un momento dado también utilizo un manejador, para pasar mensajes entre hilos.

- En el onCreate() como podrá ver, inicializo toda la interfaz de usuario. A diferencia de java, en Android además de crear los elementos visuales vía código, también pueden ser definidos mediante un XML que luego se linkará a los objetos del código. Por tanto, rescato los objetos del XML y les asigno los valores necesarios.

- Durante la ejecución del programa, necesitamos de otros programas proporcionados por Android. Necesitamos los Intents. Los Intents son objetos Android que al crearse se setean con un programa existente en el aparato, como por ejemplo la cámara que viene en el móvil. Después de ajustar los parámetros, se lanza el intent (lanzo la aplicación ajena a mi) para recuperar ciertos datos, se recuperan, y se elimina el intent.

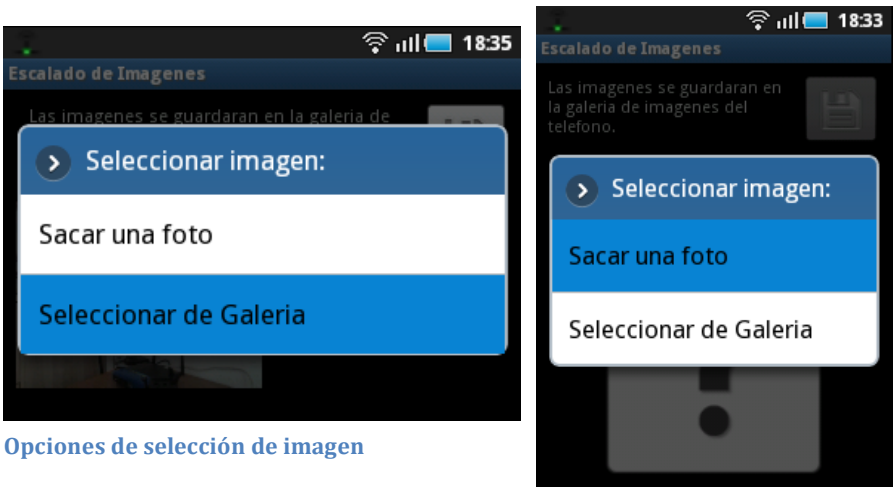
Básicamente en la pila de aplicaciones, cuando lanzo un intent, mi aplicación queda en segundo plano, el intent toma protagonismo, hace sus operaciones, y me devuelve el control a mi aplicación. Esto me permite usar el visor de fotos del fabricante en vez de usar uno programado por mi, con la consiguiente perdida de memoria y facilidad de errores. En arrancaVisor() arranco la cámara, lo mismo que en arrancaGaleria() arranco la Galeria y cuando ya tengo la foto, en onActivityResult() recupero la imagen

- Por ultimo, en el método onClick() y run(), ejecuto los algoritmos de ampliación de la imagen. Convierto la actividad en un hilo (creo un hilo nuevo con mi actividad que ya implementaba Runnable) y mediante el handler seteo los diálogos.



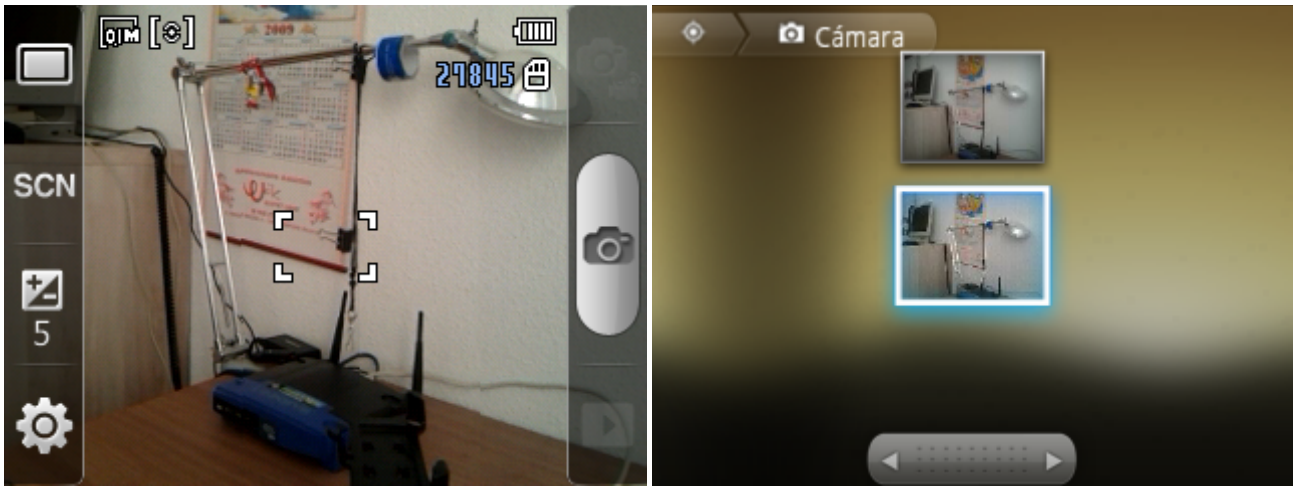
Pantalla principal en posiciones Horizontal (izquierda) y Vertical (Derecha)

Al iniciar la aplicación el usuario se encuentra esto. Para sacar una foto solo tendría que tocar el interrogante y le preguntara si quiere obtenerla de la cámara o si la tiene en la galería. En cualquiera de los dos casos se lanzaría la aplicación correspondientes, como se vera en las imágenes.



Opciones de selección de imagen

Se lanzan las aplicaciones correspondientes y se muestra la imagen donde estaba el interrogante.



Cámara de Fotos (Samsung) izquierda, y Galería derecha.

Una vez tenemos la imagen en la pantalla principal, seleccionamos el algoritmo que queremos utilizar y tocamos el botón con el icono del disquete.

Pantalla principal con la imagen sacada y dialogo de guardado en galería.

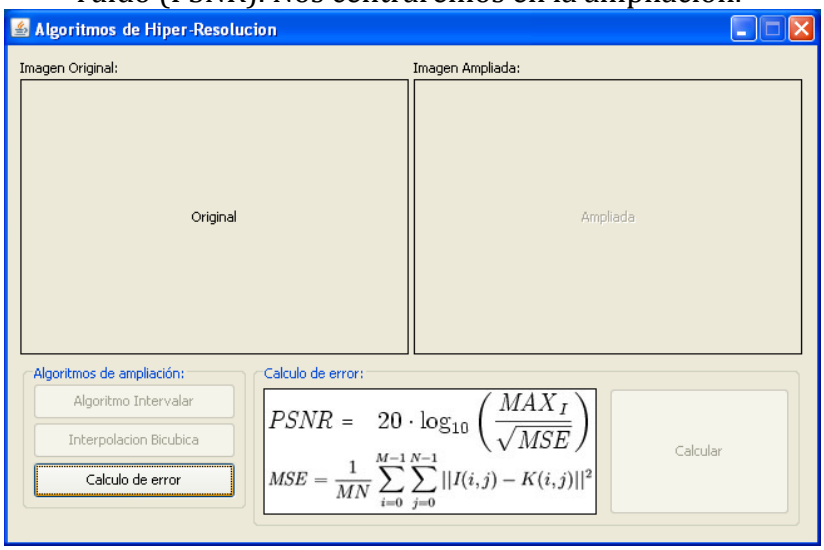
Con esto estaría mas o menos explicada la aplicación.

PROBLEMA

Durante el desarrollo del proyecto, me he dado cuenta de que por el motivo que sea, los algoritmos utilizados, o por lo menos la forma en que los he implementado, no es la mas eficiente, y por tanto Android me da errores de OutOFMemoryError, se queda sin memoria, para imágenes un poco grandes, por lo tanto en la parte de experimentos de este trabajo, hare experimentos con poca resolución en el movil, pero con mas resolución en java en el ordenador. Paso ahora a explicar brevemente el programa de escalado de imágenes en java.

PROGRAMA DE ESCALADO EN JAVA

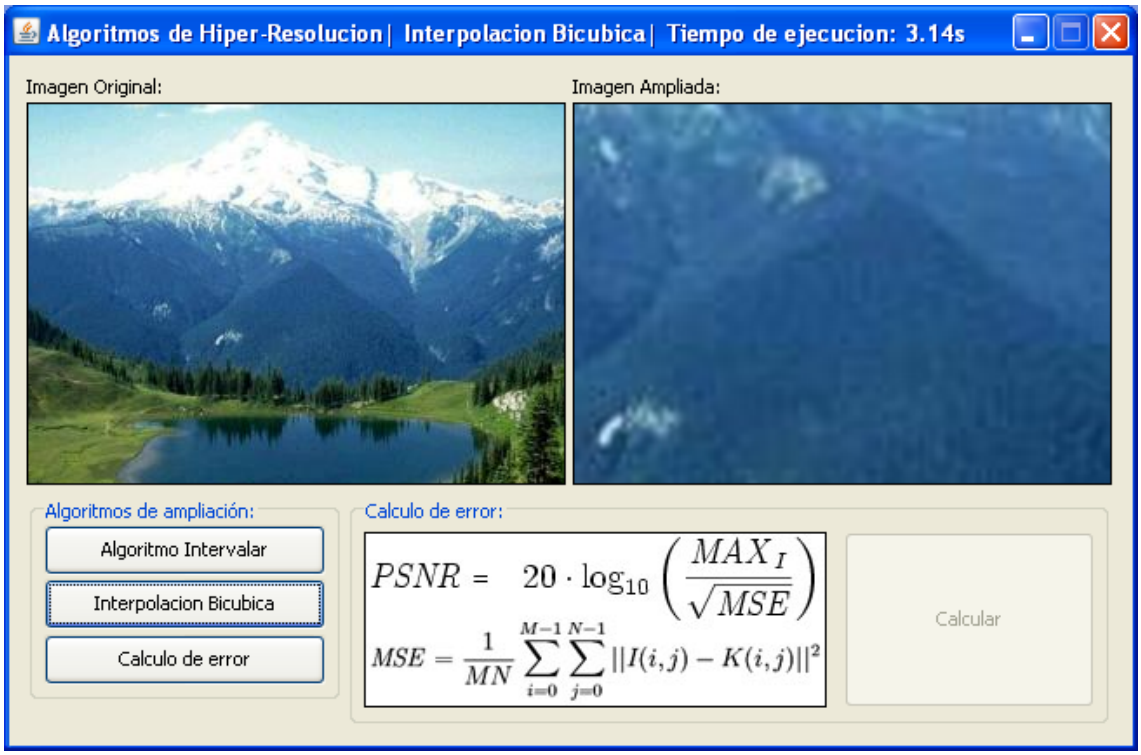
El programa se compone de dos partes, una para ampliar y otra para el calculo del ruido (PSNR). Nos centraremos en la ampliación.



Como podemos observar el la imagen tenemos dos recuadros, imagen original y ampliada. Pinchando en el original, seleccionamos la imagen original. Al seleccionar se nos activaran los dos botones de los algoritmos.

Al pinchar sobre alguno de ellos se ampliara la imagen y aparecerá en ampliada.

Al ampliarse, podremos hacer click en ampliada y asi guardar la imagen.



Para el calculo del PSNR, el procedimiento es similar, solo que seleccionamos primero el botón del calculo de error. Se activaran los dos recuadros, para seleccionar una imagen en cada uno. Seguidamente le damos a calcular.

EXPERIMENTOS

Ya hemos explicado tanto los conceptos teóricos y los algoritmos, como el funcionamiento del programa. Toca ahora, la parte practica.

Para el desarrollo de los experimentos utilizo tanto el móvil Android como una cámara de fotos. Como el tamaño de memoria que Android deja a sus aplicaciones es demasiado pequeño, las fotos que podemos ampliar en el terminal tendrán una resolución de 320x240 pixeles sin ampliar, de lo contrario tendremos un OutOf MemoryError, del que ya he hablado antes. Por tanto realizare 5 fotos en esta resolución, que ampliare a 960x720 pixeles y comparare con fotos tomadas por el mismo móvil con una resolución de 1280x960 pixeles que he escalado con la interpolación Bicúbica de Photoshop (CS5.1) a 960x720 pixeles.

Por otro lado también he tomado esas mismas fotos con una cámara de fotos con una resolución de 3456x2592 pixeles y con una resolución de 1.3 MPx escalándolas a 1152x864 (mismo proceso) para que al ampliar con java en el ordenador se puedan comparar con las de 3456x2592 pixeles de resolución.

Por tanto comparare 5 fotos con resoluciones capaces en Android y esas mismas 5 fotos con mayor resolución capaces en Java (Intel DualCore 2.16GHz 3GB RAM).

Para la comparativa empleare el PSNR (Peak Signal-to-Noise Ratio) que no deja de ser mas que un error cuadrático medio pixel a pixel con las intensidades de las dos imágenes. A nivel de desarrollo, bastaría con aplicar estas formulas:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right),$$

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}||I(i,j) - K(i,j)||^2$$

I(i,j) representa el pixel en la posición i,j de la imagen original, y k hace lo propio con la ampliada. Como veis se trata de recorrer las dos matrices, restando ambos pixeles y elevándolos al cuadrado. Estamos midiendo ruido, por lo tanto las unidades serán decibelios (db). Los valores típicos de esta operación son de 30 a 50 decibelios, siendo mayor cuando mejor es la codificación.

Para las imágenes con el ordenador solo expondré una tabla con los valores del PSNR. Para las imágenes del móvil, al final de la sección se podrán comparar tanto su PSNR como una parte a misma escala de la imagen en cuestión.

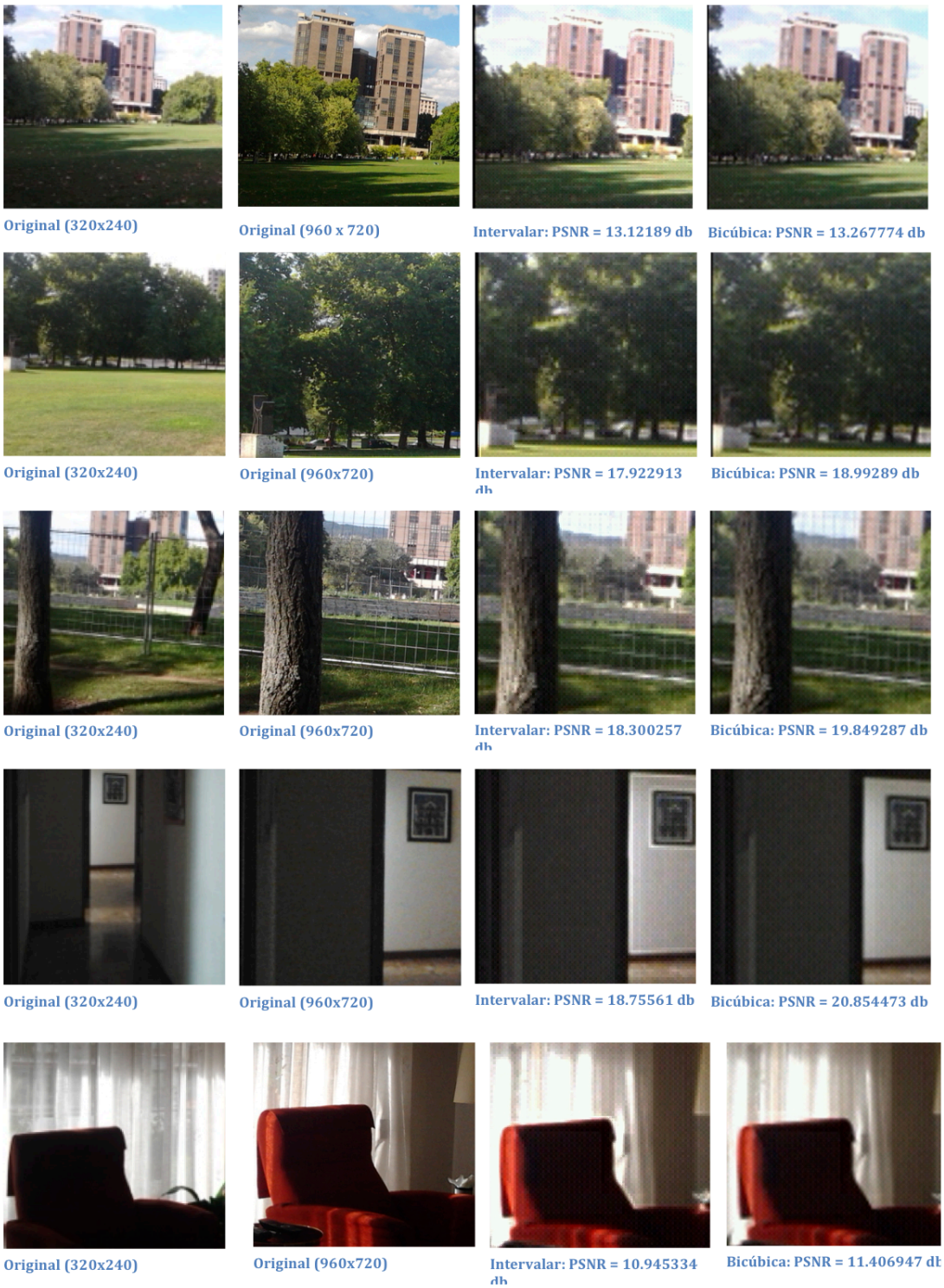
PSNR IMÁGENES CON ORDENADOR:

Imagen	Intervalar	Bicúbica
Uno	**15.253049 db**	**16.109089 db**
Dos	**13.554141 db**	**14.644458 db**
Tres	**11.073192 db**	**11.890498 db**
Cuatro	**14.094025 db**	**13.995415 db**
Cinco	**12.326473 db**	**12.428723 db**

Estos valores representan los decibelios de ruido con respecto de la imagen original (3456x2592) según las imágenes ampliadas por dichos algoritmos.

En este caso la media de las imágenes es para el caso del Intervalar 13,260176 db y para el caso de la interpolación Bicúbica 13,8136366 db.

PSNR IMÁGENES AMPLIADAS POR ANDROID:



(El orden de las imágenes indica también su nombre en la tabla de las imágenes ampliadas por ordenador)

CONCLUSIONES

Hemos explicado dos algoritmos de super-resolucion de imágenes. Pese a que el segundo tiene mas potencia, las imágenes se ven mucho mejor, el coste en memoria y ejecución es mayor que el primero, y por tanto es mas lento. Habría que hallar un compromiso entre calidad y tiempo.

El algoritmo intervalar es bastante mas rápido, pero como se puede ver en los experimentos, da mucho mejor resultado la interpolación bicúbica.

Ha sido mas fácil entender y programar el primero, ya que computacionalmente (y conceptualmente) es mas fácil de entender y programar. El segundo era muy fácil de entender conceptualmente, pero a la hora de programarlo ha tenido alguna dificultad.

PROBLEMAS A LA HORA DE DESARROLLAR

Ya he expuesto el funcionamiento de la aplicación, así como los algoritmos en que se basa, ahora expondré brevemente los problemas que he tenido (y que en general se tienen) a la hora de desarrollar aplicaciones Android.

Comencé el proyecto sin tener en mis manos un terminal físico con Android, lo que me obligo a programar y probar con el emulador de Android que ofrece Google en su SDK. Al poco de empezar, comencé a probar con la cámara, con los visores, y me di cuenta de que el emulador de Android no hace una conexión con la webcam del ordenador. En otras palabras, el emulador no tenia cámara.

Esto suponía un problema, puesto que yo necesitaba cámara para realizar el proyecto y las pruebas. En internet, siempre hay alguien que ha tenido el mismo problema que tienes tu, por tanto indagando, encontré una forma de añadirle cámara a mi emulador Android. La idea era que yo en mi maquina virtual java conectaba una webcam y servía las imágenes por un socket, después, en Android tenia una clase SocketCamera que se comportaba como la clase Camera de Android, y que recibía dichas imágenes por el socket. Ya había solucionado el primer problema.

Tras hacer funcionar la cámara, diseñe un visor, para tomar las imágenes, pero cuando lo probaba con un terminal real (HTC Desire HD de un amigo) se colgaba el programa debido a perdidas de memoria y permisos. La solución por la que me decante fue la de usar Intents. He intentado explicar antes lo que son, los detallare un poco mas. En vez de programar yo un visor, ¿por qué no uso la aplicación de la cámara que proporciona Android (Samsung en mi caso, puesto que mas tarde adquirí un terminal) para tomar las fotos?. Y eso hice. Cree un Intent que lanza la cámara y recupera la foto tomada. Ya no necesitaba la clase SocketCamera, ni su análoga en Android, ni lo expuesto antes.

Ya tenia mi aplicación funcionando. Después de esto, me percaté que cuando cambiaba la orientación del móvil, se reseteaba mi aplicación. Leí en internet que esto sucede porque al cambiar de orientación, Android termina la aplicación y la vuelve a iniciar, reinicia el ciclo de vida de la aplicación. También encontré solución a esto, con métodos proporcionados por Android, que guardan la información antes girar y la reescriben después del giro.

Después de esto no he tenido mas problemas con el desarrollo, a parte de la complejidad de desarrollo de los algoritmos en si mismo, que no es un problema, sino parte del trabajo.

FUTURO DE LA APLICACIÓN

Creo que la aplicación en si, no tiene futuro como tal, me explico, la pantalla principal así como los diálogos, no tienen mas sentido que exponer los algoritmos de ampliación de imagen. Por tanto son los algoritmos lo que tienen futuro, no la aplicación en si. El futuro de esto seria un visor de imágenes nuevo, con la posibilidad de aplicación de los algoritmos en tiempo real sobre la imagen en pantalla a modo de zoom digital. Ese el futuro que le veo.

CAMBIOS Y MEJORAS

Muchas son las mejoras que se le pueden aplicar a la aplicación, principalmente lo expuesto en el epígrafe “Futuro de la aplicación”. Convertir esta aplicación meramente educativa en algo funcional como un visor de imágenes con opciones de escalado de imágenes en tiempo real (Zoom digital).

De la aplicación y de los algoritmos, la mejora seria una optimización de memoria. No he pretendido que los algoritmos sean los mejores, sino que funcione y en Android. Faltaría entonces optimización de memoria y puesto que los ratios de ampliación para ambos algoritmos no son fijos, un marco de opciones de ampliación para la ejecución de los mismos. También se le podrían programar filtros de la media, que suavicen el resultado, sobretodo en el algoritmo Intervalar.

BIBLIOGRAFIA

Documento “Magnificación”. Departamento de automática y computación UPNA

Cubic interpolation. http://www.paulinternet.nl/?page=bicubic

Taller de aplicaciones Android CPES15:
http://www.youtube.com/watch?v=WFLnhYvzKpo
http://www.youtube.com/watch?v=DsIH-qGC21I

API Android: http://developer.android.com/sdk/index.html
Stack OverFlow: http://stackoverflow.com/

ANEXOS

MainActivity.java:

```
<< imports necesarios y definición del paquete >>

public class MainActivity extends Activity implements
OnClickListener,Runnable,Handler.Callback{

      //Necesario para la pantalla de guardado:
      RadioButton upna,bicubic;
      ImageButton save;
      boolean guardar;
      ImageView capturada;

      //Camara:
      String imageFilePath,path;
      public final int CAMERA_RESULT = 0;
      public final int ACTIVITY_SELECT_IMAGE = 1;

      //Otras cosas:
      Bitmap imagen;
      ProgressDialog dialog;
      CharSequence nombre,original;
      File imageFile;
      AlertDialog alert,memoria;

      private Handler handler;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Selecciono el layout dependiendo de la configuracion de la pantalla.
        onConfigurationChanged(getResources().getConfiguration());
        inicializarUI();
    }
    @Override
    protected void onResume() {
        super.onResume();
    }
    @Override
    protected void onPause() {
        super.onPause();
    }

    public boolean onCreateOptionsMenu(android.view.Menu menu) {
        menu.add(0,0, 0, "Sacar Foto").setIcon(R.drawable.ic_menu_camera);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item){
      //arrancaVisor();
      alert.show();
        return true;
    }
```

```
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
      savedInstanceState.putString("imageFilePath", imageFilePath);
      super.onSaveInstanceState(savedInstanceState);
    }


    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
      super.onRestoreInstanceState(savedInstanceState);
      // Llamo a onCreate para que se vuelva a setear todo...
      imageFilePath = savedInstanceState.getString("imageFilePath");
      imagen = BitmapFactory.decodeFile(imageFilePath);
      guardar = savedInstanceState.getBoolean("save");
    }


    private void inicializarUI(){
      //Dialogos
      final CharSequence[] items = {"Sacar una foto", "Seleccionar de Galeria"};
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Seleccionar imagen:");
            builder.setItems(items, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int item) {
                    if(items[item]==items[0]) arrancaVisor();
                    else arrancaGaleria();
                }
            });
            alert = builder.create();
            memoria = ((new AlertDialog.Builder(this)).setMessage("Seleccione fotos de menor
tamaño.").setTitle("Error de memoria:")).create();

            //El manejador, para la comunicacion entre hilos.
            handler = new Handler(this);

        //Bloqueamos los sensores de orientacion:
        //0: Landscape
        //1: Portrait
        setTitle("Escalado de Imagenes");

        //Recuperamos los elementos del display:
            upna = (RadioButton) findViewById(R.id.upna);
            bicubic = (RadioButton) findViewById(R.id.bicubic);
            capturada = (ImageView) findViewById(R.id.capturada);
            save = (ImageButton) findViewById(R.id.save);

            //Seteamos variables:
            imageFilePath  =  Environment.getExternalStorageDirectory().getAbsolutePath()  +
"/tmp_image.jpg";
            path     =     Environment.getExternalStorageDirectory().getAbsolutePath()     +
"/DCIM/Camera/";
            save.setEnabled(false);
            save.setOnClickListener(this);
            capturada.setOnClickListener(this);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
```

```
        // Checks the orientation of the screen
        if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            setContentView(R.layout.landscape);
        } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
            setContentView(R.layout.portrait);
        }
        inicializarUI();
        if(imagen != null){
            capturada.setImageBitmap(imagen);
            save.setEnabled(true);
        }
    }


    private void arrancaVisor(){
      //Preparo y lanzo la camara:
      imageFile = new File(imageFilePath);
            Uri imageFileUri = Uri.fromFile(imageFile);
            Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
            i.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, imageFileUri);

      //i.putExtra(MediaStore.EXTRA_SCREEN_ORIENTATION,ActivityInfo.SCREEN_ORIENTATION_NOSEN
SOR);
            startActivityForResult(i, CAMERA_RESULT);
    }

    private void arrancaGaleria(){
      Intent            i            =            new            Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
      startActivityForResult(i, ACTIVITY_SELECT_IMAGE);
    }


    protected void onActivityResult(int requestCode, int resultCode, Intent intent){
            super.onActivityResult(requestCode, resultCode, intent);
            if (requestCode== CAMERA_RESULT && resultCode == RESULT_OK)      {
                  imagen = BitmapFactory.decodeFile(imageFilePath);
                  capturada.setImageBitmap(imagen);
                  //imageFile.delete();
                  save.setEnabled(true);
            }else if(requestCode == ACTIVITY_SELECT_IMAGE && resultCode == RESULT_OK){
                  Uri selectedImage = intent.getData();
            String[] filePathColumn = {MediaStore.Images.Media.DATA};

            Cursor cursor = getContentResolver().query(selectedImage, filePathColumn, null,
null, null);
            cursor.moveToFirst();

            int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
            String filePath = cursor.getString(columnIndex);
            cursor.close();

            imagen = BitmapFactory.decodeFile(filePath);
            capturada.setImageBitmap(imagen);
            save.setEnabled(true);
            }
      }
```

```
	@Override
	public void onClick(View elemento) {
		if(elemento == capturada && imagen == null){
			alert.show();
		}else if(elemento == save){
			//Hacemos el algoritmo en otro hilo:
			//Principalmente por el tema del dialog.
			new Thread(this).start();
		}
	}

	public void run(){
		handler.sendEmptyMessage(0);
		try{
			if(upna.isChecked()){
				nombre = System.currentTimeMillis() + "_Intervalar_HD";
				saveBitmap((new	Intervalar(imagen)).getScaledImage(),	path,
nombre+".jpeg");
				MediaStore.Images.Media.insertImage(getContentResolver(),
path+nombre+".jpeg", (String) nombre, (String) null);
			}else if(bicubic.isChecked()){
				nombre = System.currentTimeMillis() + "_Bicubic_HD";
				saveBitmap((new	Bicubic(imagen)).getScaledImage(),	path,
nombre+".jpeg");
				MediaStore.Images.Media.insertImage(getContentResolver(),
path+nombre+".jpeg", (String) nombre, (String) null);
			}
		}catch(FileNotFoundException ex){
		}catch(OutOfMemoryError ex){
			handler.sendEmptyMessage(2);
		}
		//Paramos el dialogo.
	  handler.sendEmptyMessage(1);
	}

	public boolean saveBitmap(Bitmap img, String path, String name){
		ByteArrayOutputStream bytes = new ByteArrayOutputStream();
		img.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
		//you can create a new file name "test.jpg" in sdcard folder.
		File f = new File(path + name);
		//write the bytes in file
		FileOutputStream fo;
		try {
			f.createNewFile();
			fo = new FileOutputStream(f);
			fo.write(bytes.toByteArray());
			return true;
		} catch (Exception e) {
			e.printStackTrace();
			return false;
		}
	}
	@Override
	public boolean handleMessage(Message msg) {
		switch(msg.what){
		case 2:	dialog.dismiss();
			imagen.recycle();
			imagen = null;
			Resources res = getResources();
```

```
                        capturada.setImageDrawable(res.getDrawable(R.drawable.unimage));
                        memoria.show();
                        break;
                case 1: dialog.dismiss();

        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD);
                    getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
                    getWindow().clearFlags(WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
                    break;
                case 0:
                        dialog = new ProgressDialog(this);
                        dialog.setMessage("Escalando y guardando en galeria ...");
                        dialog.setTitle("Procesando:");
                        dialog.show();

        getWindow().addFlags(WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD);
                    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
                    getWindow().addFlags(WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
                    break;
                }
                return true;
        }
}
```

Intervalar.java:

```
package com.hkf.camera;

import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;


/*
 * Implementacion del agoritmo de magnificacion del Laboratorio:
 */
//Uso metodo round de libreria Math. (tambien existe en Android)
public class Intervalar{
	short width,height;
	int[][] img;
	Config configuracion;

	Intervalar(Bitmap imagen) {
		img = getMatrix(imagen);
		configuracion = imagen.getConfig();
		width = (short) imagen.getWidth();
		height = (short) imagen.getHeight();
		//imagen.recycle();
	}

	protected int[][] getMatrix(Bitmap original){
		int height = original.getHeight();
		int width = original.getWidth();
		int matrix[][] = new int[width][height];
		for(int i = 1; i< width; i++)
			for(int j = 1; j< height; j++){
				matrix[i][j] = original.getPixel(i,j);
			}
		return matrix;
	}


	//Calculo de la desviacion tipica de la foto:
	protected float getDeviation(short[][] matrix){
		//float[][] normalizada = t.getNormalizedMatrix(matrix);
		float actual;
		float media = Tools.avg(Tools.getNormalizedMatrix(matrix));
		float var = 0;
		for(int i = 1; i< width; i++)
			for(int j = 1; j< height; j++){
				actual = matrix[i][j]/255;
				var+=(actual-media)*(actual-media);
			}
		var/=width*height;
		var=(float)Math.sqrt(var);
		return var;
	}

	public short[][] getScaledMatrix(float sigma,short[][] matriz){
		int tam = 3;
		float nuevamatriz[][] = new float[width*tam][height*tam];
		float ventana[][] = new float[tam][tam], nueva[][] = new float[tam][tam];
		@SuppressWarnings("unused")
		double W,a,b;
		int n = 0,m = 0;
```

```
            int gi=1,gj=1;
            for(int i = 0; i<= width -1; i++){
                  for(int j = 0; j<= height-1; j++){
                        for(n=0; n<tam; n++)
                              for(m=0; m<tam; m++){
                                    try{
                                    ventana[n][m] = (float) matriz[(i+n)-1][(j+m)-1]/255;
                                    }catch(ArrayIndexOutOfBoundsException ex){
                                          ventana[n][m] = (float) matriz[i][j]/255;
                                    }
                              }
                        /*
                         * La funcion  del intervalo es:
                         *    inter(pixel[i][j],    sigma*W)    =    [pixel[i][j]*(1-sigma*W),
pixel[i][j]*(1-sigma*W)+(sigma*W)]
                         */
                        W = (Tools.max(ventana)- Tools.min(ventana));
                        a = ((double) matriz[i][j]/255)*(1-sigma*W);
                        b = a + (sigma*W);
                        //Tenemos que hacer un recorrido y crear una nueva ventana.
                        for(n=0; n<3; n++)
                              for(m=0; m<3; m++)
                                    nueva[n][m] = (float)(a + ventana[n][m]* W);
                        /* En la nueva matriz, copiamos las ventanas como "macroPixels". */
                        for(n=0;n<3 && (n+gi)<(width*3)-1;n++)
                              for(m=0;m<3 && (m+gj)<(height*3)-1;m++){
                                    nuevamatriz[n+gi][m+gj] = nueva[n][m];
                              }
                        gj+=3;
                  }
                  gi+=3;
                  gj = 0;
            }
            return Tools.getStandardMatrix(nuevamatriz);
      }

      public Bitmap getScaledImage(){
            short[][]                                    sred                                    =
getScaledMatrix(getDeviation(Tools.redMatrix(img)),Tools.redMatrix(img));
            short[][]                                   sgreen                                   =
getScaledMatrix(getDeviation(Tools.greenMatrix(img)),Tools.greenMatrix(img));
            short[][]                                    sblue                                   =
getScaledMatrix(getDeviation(Tools.blueMatrix(img)),Tools.blueMatrix(img));
            Bitmap imagen = Bitmap.createBitmap(sred.length,sred[0].length, configuracion);
            for(int i = 1; i< sred.length; i++)
                  for(int j = 1; j< sred[0].length; j++){
                        imagen.setPixel(i,    j,    Tools.getRGB(sred[i][j],    sgreen[i][j],
sblue[i][j]));
                  }
            return imagen;
      }

}
```

Bicubic.java:

```
package com.hkf.camera;

import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;

public class Bicubic{
	short width,height;
	int[][] img;
	Config configuracion;


	Bicubic(Bitmap imagen) {
		img = getMatrix(imagen);
		configuracion = imagen.getConfig();
		width = (short) imagen.getWidth();
		height = (short) imagen.getHeight();
		//imagen.recycle();
	}

	protected int[][] getMatrix(Bitmap original){
		int height = original.getHeight();
		int width = original.getWidth();
		int matrix[][] = new int[width][height];
		for(int i = 1; i< width; i++)
			for(int j = 1; j< height; j++){
				matrix[i][j] = original.getPixel(i,j);
			}
		return matrix;
	}

	public int bicubic (int[] p, float x) {
		return (int) (p[1] + 0.5 * x*(p[2] - p[0] + x*(2.0*p[0] - 5.0*p[1] + 4.0*p[2] -
p[3] + x*(3.0*(p[1] - p[2]) + p[3] - p[0]))));
	}

	public short[][] getScaledMatrix(float sigma,short[][] matrix){
		// x,y coords for pixels in old and new imagesfloat ox,oy;
		int   nx,ny,ox,oy;
		float oxf,oyf;
		int actual;
		short[][] nimg = new short[(int) (width*sigma)][(int) (height*sigma)];
	int[] uno = new int[4],dos=new int[4], tres = new int[4], cuatro = new int[4],t = new
int[4];
		float dx = (float)(1/sigma);
		float dy = (float)(1/sigma);

		for (ny = 0; ny < height*sigma; ny++){
			oy = (int) (ny * dy);
			oyf= (float)((dy*ny)-oy);
			for (nx = 0; nx < width*sigma; nx++) {
				ox = (int) (nx * dx);
		   //Posicion entera respecto de la imagen antigua.
				oxf= (float)((dx*nx)-ox);
		   // Los decimales de volver de la nueva posicion a la antigua. Nuevos puntos.

				//Si por algun motivo se salta, ponemos el pixel ox,oy
				for(int k = 0; k<=3; k++){
					try{
```

```
                                    uno[k] = matrix[(ox-1)+k][oy-1];
                              }catch(ArrayIndexOutOfBoundsException ex){
                              uno[k] = matrix[ox][oy];
                              }
                              try{
                                    dos[k] = matrix[(ox-1)+k][oy];
                              }catch(ArrayIndexOutOfBoundsException ex){
                                    dos[k] = matrix[ox][oy];
                              }
                              try{
                                    tres[k] = matrix[(ox-1)+k][oy+1];
                              }catch(ArrayIndexOutOfBoundsException ex){
                                    tres[k] = matrix[ox][oy];
                              }
                              try{
                                    cuatro[k] = matrix[(ox-1)+k][oy+2];
                              }catch(ArrayIndexOutOfBoundsException ex){
                                    cuatro[k] = matrix[ox][oy];
                              }
                        }
                        t[0] = bicubic(uno, oxf);
                        t[1] = bicubic(dos, oxf);
                        t[2] = bicubic(tres, oxf);
                        t[3] = bicubic(cuatro, oxf);

                        //Interpolo los resultados;
                        actual = bicubic(t, oyf);
                        if (actual>255) nimg[nx][ny] = 255;
                        else if(actual<0) nimg[nx][ny] = 0;
                        else nimg[nx][ny] = (short) actual;
                  }
            }
            return nimg;
      }

      public Bitmap getScaledImage(){
            float amp = 3;
            short[][] sred = getScaledMatrix(amp,Tools.redMatrix(img));
            short[][] sgreen = getScaledMatrix(amp,Tools.greenMatrix(img));
            short[][] sblue = getScaledMatrix(amp,Tools.blueMatrix(img));
            Bitmap imagen = Bitmap.createBitmap(sred.length,sred[0].length, configuracion);
            for(int i = 1; i< sred.length; i++)
                  for(int j = 1; j< sred[0].length; j++){
                        imagen.setPixel(i,    j,    Tools.getRGB(sred[i][j],    sgreen[i][j],
sblue[i][j]));
                  }
            return imagen;
      }

}
```

Tools.java: (Clase "Estatica" auxiliar con métodos "Herramienta")

```
package com.hkf.camera;

//Clase Estatica de Herramientas.

public class Tools {
	//Devuelvo la matriz Verde:
	public static short[][] greenMatrix(int[][] matrix){
		short[][] green = new short[matrix.length][matrix[0].length];
		for(int i = 1; i< matrix.length; i++)
			for(int j = 1; j< matrix[0].length; j++){
				green[i][j] = getGreen(matrix[i][j]);
			}
		return green;
	}
	//Devuelvo la matriz Roja:
	public static short[][] redMatrix(int[][] matrix){
		short[][] red = new short[matrix.length][matrix[0].length];
		for(int i = 1; i< matrix.length; i++)
			for(int j = 1; j< matrix[0].length; j++){
				red[i][j] = getRed(matrix[i][j]);
			}
		return red;
	}
	//Devuelvo la matriz Azul:
	public static short[][] blueMatrix(int[][] matrix){
		short[][] blue = new short[matrix.length][matrix[0].length];
		for(int i = 1; i< matrix.length; i++)
			for(int j = 1; j< matrix[0].length; j++){
				blue[i][j] = getBlue(matrix[i][j]);
			}
		return blue;
	}
	//Normalizamos una matriz:
	public static float[][] getNormalizedMatrix(short[][] matrix){
		float[][] salida = new float[matrix.length][matrix[0].length];
		for(int i = 1; i< matrix.length; i++)
			for(int j = 1; j< matrix[0].length; j++)  salida[i][j] = matrix[i][j]/255;
		return salida;
	}
	//"Des"normalizamos una matriz:
	public static short[][] getStandardMatrix(float[][] matrix){
		short[][] salida = new short[matrix.length][matrix[0].length];
		short actual;
		for(int i = 1; i< matrix.length; i++)
			for(int j = 1; j< matrix[0].length; j++){
				actual = (short) Math.round(matrix[i][j] * 255);
				if (actual>255) salida[i][j] = 255;
				else if(actual<0) salida[i][j] = 0;
				else salida[i][j] = actual;
			}
		return salida;
	}

	//Metodos maximo y minimo sobre el intervalo 0:1
	public static float max(float[][] ventana){
		float maximo=0;
		for(int i=0; i<ventana.length; i++)
			for(int j=0; j<ventana[0].length; j++)
```

```
                        if(maximo<ventana[i][j]     &&     ventana[i][j]!=-1)     maximo     =
ventana[i][j];
            return maximo;
      }
      public static float min(float[][] ventana){
            float minimo=1;
            for(int i=0; i<ventana.length; i++)
                  for(int j=0; j<ventana[0].length; j++)
                        if(minimo>ventana[i][j]     &&     ventana[i][j]!=-1)     minimo     =
ventana[i][j];
            return minimo;
      }
      public static float avg(float[][] ventana){
            float contador=0;
            int cuenta=0;
            for(int i=0; i<ventana.length; i++)
                  for(int j=0; j<ventana[0].length; j++){
                        contador+=ventana[i][j];
                        cuenta++;
                  }
            return contador/cuenta;
      }
      //Como todas las implementaciones van a usarlas, las dejo en la superclase:
      // Yo trabajo con valores entre 0:255, pero el necesita enteros.
      public static int getRGB(short red, short green, short blue){
            int rgb = red;
            rgb = ((rgb << 8) + green);
            rgb = ((rgb << 8) + blue);
            return rgb;
      }
      public static short getGreen(int rgb){
            return (short) ((rgb & 0x0000ff00) >> 8);
      }
      public static short getBlue(int rgb){
            return (short) (rgb & 0x000000ff);
      }
      public static short getRed(int rgb){
            return (short) ((rgb & 0x00ff0000) >> 16);
      }
}
```

PSNR.java:

```
package aplicacion;

import java.awt.image.BufferedImage;


public class PSNR {

	private final static int RED = 0;
	private final static int GREEN = 1;
	private final static int BLUE = 2;

	public static float getValue(BufferedImage original, BufferedImage ampliada) throws
ArithmeticException {
		float total = 0;
		int[][] actualo,actuala;
		for(int i = 0; i<=2; i++){
			actualo = getSubMatrix(getMatrix(original),i);
			actuala = getSubMatrix(getMatrix(ampliada),i);
			total += getColorValue(actualo,actuala);
		}
		return total/3;
	}

	//El valor mas alto que puede tomar un pixel es 255;
	private static float getColorValue(int[][] original, int[][] ampliada) throws
ArithmeticException {
		float MSE = getMSEValue(original,ampliada);
		return (float) (20 * Math.log10(255/Math.sqrt(MSE)));
	}

	private static float getMSEValue(int[][] original, int[][] ampliada){
		float contador=0;
		for(int i = 0; i<original.length; i++)
			for(int j = 0; j<original[0].length ; j++){
				contador += (ampliada[i][j] - original[i][j])*(ampliada[i][j] -
original[i][j]);
				//System.out.println("Contador: "+contador+", ampliada: "+
ampliada[i][j]+", original: "+original[i][j]);
			}
		return contador / (original.length*original[0].length);
	}

	protected static int max(int[][] ventana){
		int maximo=0;
		for(int i=0; i<ventana.length; i++)
			for(int j=0; j<ventana[0].length; j++)
				if(maximo<ventana[i][j] && ventana[i][j]!=-1) maximo =
ventana[i][j];
		return maximo;
	}

	protected static double[][] getNormalizedMatrix(int[][] matrix){
		int tw = matrix.length,th=matrix[0].length;
		double[][] salida = new double[tw][th];
		for(int i = 1; i< tw; i++)
			for(int j = 1; j< th; j++) salida[i][j] = matrix[i][j]/255;
		return salida;
	}
```

```
	private static int[][] getMatrix(BufferedImage original){
		int height = original.getHeight();
		int width = original.getWidth();
		int matrix[][] = new int[width][height];
		for(int i = 1; i< width; i++)
			for(int j = 1; j< height; j++)
				matrix[i][j] = original.getRGB(i,j);
		return matrix;
	}

	private static int[][] getSubMatrix(int[][] matrix,int color){
		int width = matrix.length, height = matrix[0].length;
		int[][] matriz = new int[width][height];
		for(int i = 1; i< width; i++)
			for(int j = 1; j< height; j++){
				switch(color){
				case RED:   matriz[i][j] = getRed(matrix[i][j]);
						break;
				case GREEN: matriz[i][j] = getGreen(matrix[i][j]);
						break;
				case BLUE:  matriz[i][j] = getBlue(matrix[i][j]);
				}
			}
		return matriz;
	}

	public static int getRGB(int red, int green, int blue){
		int rgb = red;
		rgb = (rgb << 8) + green;
		rgb = (rgb << 8) + blue;
		return rgb;
	}
	public static int getGreen(int rgb){
		return (rgb & 0x0000ff00) >> 8;
	}
	public static int getBlue(int rgb){
		return rgb & 0x000000ff;
	}
	public static int getRed(int rgb){
		return (rgb & 0x00ff0000) >> 16;
	}
}
```