2 DESCRIPCIÓN DEL SISTEMA 'VISUAL'

VISUAL es un programa gráfico que se ejecuta bajo MS® Windows (95/98/Me/XP) y que tiene como objetivo la especificación gráfica y la simulación de algoritmos de Visión Artificial. Está formado por un entorno gráfico y por un conjunto de operaciones de Visión Artificial que se integran dentro del entorno.

Las operaciones de Visión Artificial que se integran en VISUAL se denominan *OPIs* (Objetos de Procesamiento de Imágenes). Los OPIs son módulos de código externos al entorno gráfico (módulos EXE -ejecutables- o módulos DLL -librerías de enlace dinámico-). Esto permite desarrollar cualquier programa en el lenguaje deseado para realizar una cierta operación e incorporarlo al entorno como un nuevo OPI sin necesidad de alterar la interfaz gráfica de VISUAL. De este modo, podemos ampliar las funcionalidades de VISUAL de manera sencilla y transparente. Prueba de ello es que todas las operaciones que se han desarrollado en este proyecto se han incorporado a VISUAL sin necesidad de modificar el código fuente del entorno gráfico.

Cada OPI se corresponde con un único módulo de código pero puede realizar una o más funciones de procesamiento dependiendo de su tipo. Así, por ejemplo, el OPI de operaciones aritméticas (fichero DLL *aritmeticas.dll*) contiene diversas funciones: diferencia, división, ínfimo, media, media saturación, producto, resta, suma y supremo. Cada OPI tiene asociado un conjunto de propiedades propias que serán compartidas por todas sus funciones y que serán establecidas por el usuario en el momento de la ejecución de cada función mediante un diálogo. Por ello, a la hora de decidir qué funciones se implementan en un mismo OPI tendremos que tener en cuenta que todas recibirán los mismos tipos de parámetros de configuración.

Otro concepto fundamental de VISUAL es el *esquema*. Un esquema es una representación gráfica de un algoritmo de Visión Artificial. Estará formado por un conjunto de OPIs interconectados de tal manera que las salidas de unos OPIs se conectarán a las entradas de otros, representando un flujo de datos. VISUAL permite guardar los esquemas generados en ficheros con extensión *esq*. En la siguiente figura mostramos la representación gráfica dentro de VISUAL de un esquema.

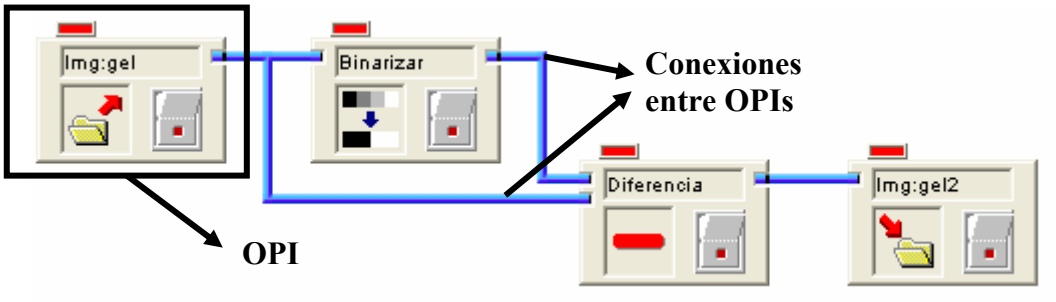

Figura 2.1. Ejemplo de un esquema en VISUAL. Está formado por OPIs (rectángulos grises) y conexiones entre OPIs (líneas azules).

2.1 Guía para el usuario de VISUAL

Con este apartado pretendemos describir el funcionamiento del entorno VISUAL desde el punto de vista de un usuario que desea utilizarlo para diseñar y probar algoritmos de Visión Artificial. No pretendemos describir todas las opciones del programa de manera detallada sino que sólo enumeraremos aquellas que son útiles para probar las operaciones desarrolladas en este proyecto.

2.1.1 Instalación y configuración de VISUAL

Para poder utilizar VISUAL en nuestro ordenador debemos tener instalada alguna versión del sistema operativo MS® Windows (en nuestro caso hemos utilizado MS® Windows XP Profesional con Service Pack 2).

Para instalar VISUAL en nuestro ordenador tenemos que copiar la carpeta *Visual* del CD-ROM que se adjunta con este documento en la ubicación que deseemos de nuestro disco duro (P. Ej.: en el directorio raíz de la partición C:). Para ejecutar VISUAL sólo habrá que ejecutar el fichero *visual32.exe* que se encuentra dentro de la carpeta copiada al disco duro.

La primera vez que ejecutemos VISUAL, aparecerá un diálogo de configuración (figura 2.2) que tendremos que completar con la localización de los principales directorios y archivos utilizados por el programa:

- Archivo de descripción de módulos: Debemos indicar la ruta del fichero *modulos.ini*. Este fichero de texto contendrá un listado con todos los OPIs y funciones accesibles desde VISUAL. En el apartado 2.2 describiremos el formato de este fichero en detalle.
- Directorio raíz para archivos de módulos: Este directorio contendrá todos los módulos binarios externos (ficheros EXE o DLL) que implementan alguna operación de Visión Artificial para VISUAL. En el apartado 2.3 describiremos en detalle la forma de desarrollar estos módulos.
- Directorio para archivos de imágenes: Este directorio contiene los iconos de cada una de las funciones implementadas en los módulos. Los iconos serán imágenes Dib-Bitmap (BMP) de 32x32 píxeles y 16 colores sin compresión. Estos iconos se mostrarán en todos los OPIs dibujados en los esquemas gráficos de VISUAL y permitirán diferenciar visualmente unos OPIs de otros.
- Directorio para archivos de interfaz: Por cada módulo externo (EXE o DLL) habrá un fichero de interfaz con el mismo nombre pero con extensión *ini*. Cada fichero de interfaz describirá el formato del cuadro de diálogo a través del que el usuario inicializa las propiedades del OPI correspondiente. Este directorio también contendrá ficheros de texto *txt* con los mensajes de error de cada OPI.
- Directorio temporal para intercambio: Servirá como lugar de almacenamiento temporal de las imágenes intermedias generadas por un esquema de VISUAL. Las imágenes intermedias serán eliminadas automáticamente cuando termine la ejecución del esquema correspondiente.

- Directorio para los archivos de ayuda de los módulos: Este directorio contiene a su vez varios subdirectorios con los ficheros de ayuda HMTL correspondientes a cada OPI. Estos ficheros se visualizarán cuando el usuario solicite ayuda sobre un OPI dentro del entorno VISUAL.

En la siguiente figura mostramos el diálogo de configuración de directorios de VISUAL (con la configuración correcta si la carpeta *Visual* se ha copiado en el directorio raíz de C:). Esta configuración se podrá cambiar en cualquier momento volviendo a mostrar este diálogo con el menú "*Archivo/Configuración de Directorios…"* de VISUAL. Este diálogo volverá a aparecer también si VISUAL detecta al arrancar que alguno de los directorios o archivos indicados no existen.

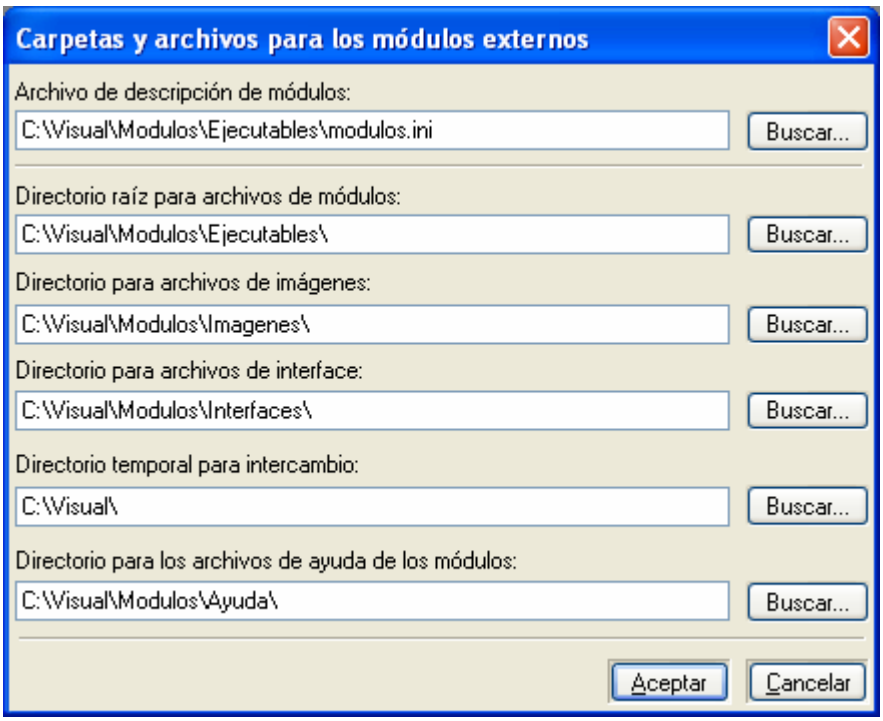

Figura 2.2. Diálogo de Configuración de Directorios en VISUAL. Con este diálogo se indicará a VISUAL la ruta de los principales directorios y archivos necesarios para su funcionamiento.

Esta configuración de directorios se almacena en el Registro de Windows (en la clave *HKEY_CURRENT_USER\Software\VisualOPI*). De este modo, una vez establecida esta configuración ya no será necesario volver a introducirla si no se cambia la localización de alguno de los directorios. Sin embargo, al guardar esta información en el Registro, un usuario no podrá tener instalados varios entornos VISUAL al mismo tiempo ya que todos usarían la misma configuración. No obstante, sí será posible tener varios entornos VISUAL en la misma máquina si son instalados por distintos usuarios ya que la clave del Registro donde se almacena la configuración de directorios es propia de cada usuario de la máquina.

2.1.2 Partes del entorno gráfico VISUAL

Cuando arrancamos VISUAL (fichero *visual32.exe*), se abre el entorno gráfico mostrado en la siguiente figura.

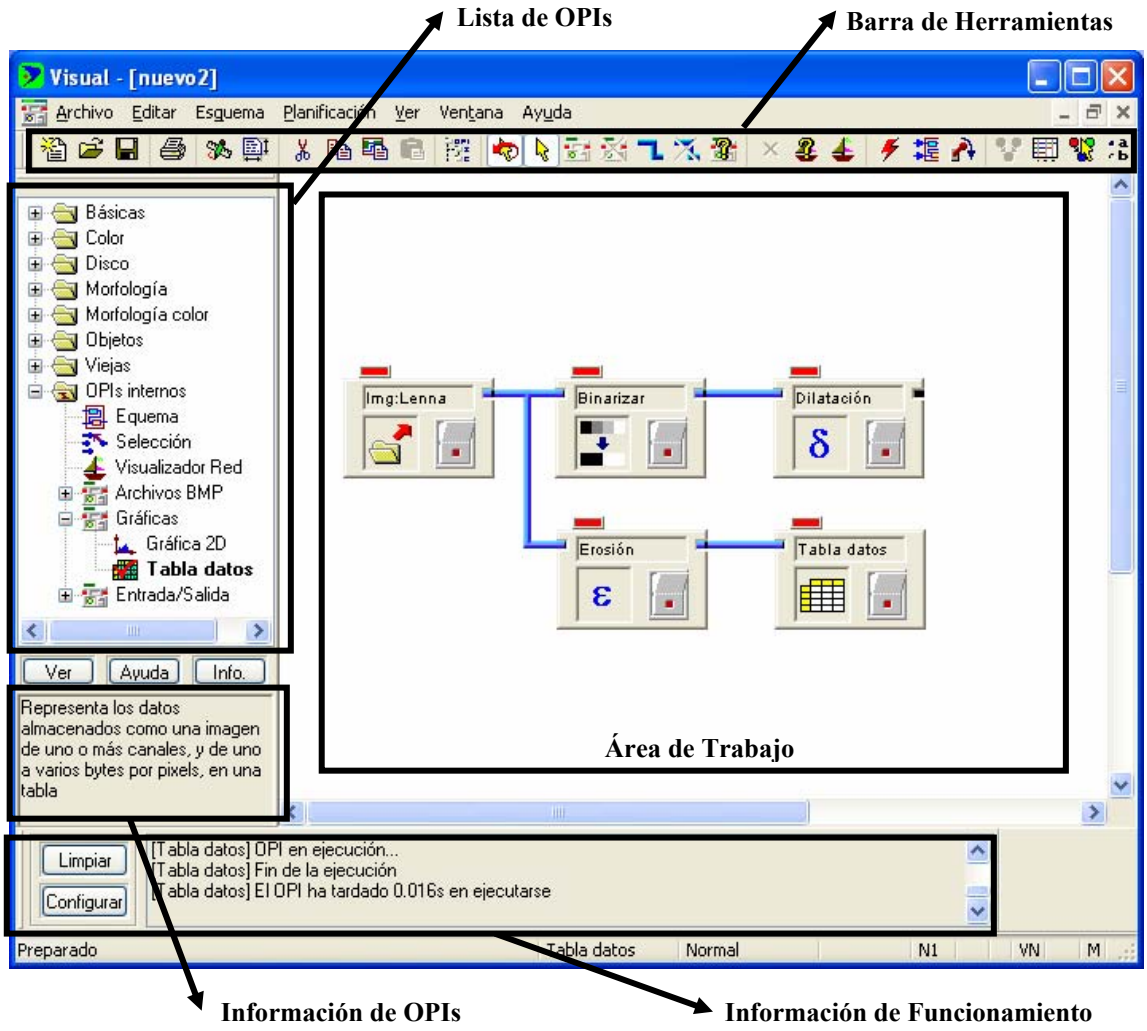

Figura 2.3. Entorno gráfico de VISUAL. Las partes más importantes de este entorno se encuentran marcadas con rectángulos negros.

Tal como se indica en la figura 2.3, las partes más importantes del entorno gráfico de VISUAL son las siguientes:

- Barra de Herramientas: Contiene las herramientas necesarias para crear esquemas con VISUAL. Describiremos las opciones más importantes de esta barra en el siguiente apartado.
- Lista de OPIs: Es una lista en forma de árbol con todos los OPIs y funciones accesibles desde VISUAL. Los niveles superiores (identificados mediante un icono de una carpeta) representan clases que sirven para organizar los OPIs por tipos: Básicas, Color, Disco, Morfología, Morfología color, Objetos, Viejas y OPIs internos. Los niveles intermedios (identificados mediante un icono de un OPI) representan los OPIs (módulos externos) accesibles desde VISUAL. Los niveles inferiores (identificados mediante un icono propio o un

círculo azul) representan funciones de los OPIs de los que cuelgan. Cuando se haga doble clic sobre una función, ésta se marcará como seleccionada mediante un tick rojo y su texto se pondrá en negrita (P. Ej.: En la figura 2.3 podemos ver que se ha seleccionado la función "Tabla datos"). La función seleccionada será la que se añada al esquema mostrado en el área de trabajo.

- Información de OPIs: En este recuadro se mostrará una breve descripción del OPI seleccionado en la lista de OPIs.
- Área de Trabajo: En esta parte se mostrarán los esquemas en los que se está trabajando. Se podrán tener abiertos varios esquemas en distintas ventanas, contenidas dentro del área de trabajo.
- Información de Funcionamiento: En este recuadro se mostrará información (mensajes de error, tiempos de ejecución…) sobre los OPIs que se están ejecutando en el entorno.

2.1.3 Barra de herramientas de VISUAL

En la siguiente tabla describimos las opciones de la barra de herramientas que nos serán más útiles para probar las operaciones desarrolladas en este proyecto. También indicaremos los menús equivalentes a cada herramienta.

Herramienta (Icono/Menú)	Descripción
Nuevo (Archivo/Nuevo…)	Crea un esquema nuevo vacío (fichero *esq*) y lo muestra en el área de trabajo.
Abrir (Archivo/Abrir…)	Abre un esquema existente (fichero *esq*) mostrando un diálogo para seleccionar su ruta.
Guardar (Archivo/Guardar)	Guarda el esquema actualmente seleccionado en disco.
Imprimir (Archivo/Imprimir)	Imprime el esquema seleccionado. Se podrá ver cómo quedará el documento impreso con la opción *"Presentación Preliminar"* del menú *Archivo*. También podremos configurar la impresora con la opción *"Configurar Impresora…"* del menú *Archivo*.
Redibujar (Ventana/Redibujar Esquema)	Redibuja el esquema seleccionado.
Propiedades (Esquema/Propiedades…)	Muestra un diálogo con las propiedades (formato de la página) del esquema.
Cortar OPI (Editar/Cortar OPI)	Cambia el modo de edición a "*modo cortar*" (cursor en forma de tijeras). Cada vez que se haga clic sobre un OPI, se cortará y se copiará en el Portapapeles de VISUAL.
Copiar OPI (Editar/Copiar OPI)	Cambia el modo de edición a *"modo copiar"* (cursor en forma de dos hojas). Cada vez que se haga clic sobre un OPI, se copiará al Portapapeles de VISUAL.
Copiar Esquema (Esquema/Copiar como Imagen)	Copia un esquema en el Portapapeles de Windows como una imagen para poder copiarlo en otros programas externos.
Pegar OPI (Editar/Pegar OPI)	Pega el último OPI almacenado en el Portapapeles de VISUAL.

Herramienta (Icono/Menú)	Descripción
 **Mostrar Selección** (Ver/Mostrar Selección)	Expande el árbol de OPIs para mostrar el OPI seleccionado por el usuario.
 **Vuelta a Modo Normal** (Edición/Vuelta a normal)	Cuando esté activa esta opción, cada vez que se realice una operación sobre el esquema se volverá automáticamente al modo de edición *"normal"*.
 **Modo Normal** (Edición/Edición Normal)	Cambia el modo de edición a *"modo normal"* (cursor en forma de cruz). Cada vez que se haga clic sobre un OPI, se podrá mover por el esquema mientras se mantenga pulsado el botón izquierdo del ratón.
 **Añadir OPI** (Edición/Añadir OPI…)	Cambia el modo de edición a *"modo añadir OPI"* (cursor en forma de OPI). Cada vez que se haga clic sobre el esquema, se añadirá un OPI del tipo seleccionado en la lista de OPIs.
 **Quitar OPI** (Edición/Quitar OPI)	Cambia el modo de edición a *"modo quitar OPI"* (cursor en forma de OPI con borrador). Cada vez que se haga clic sobre un OPI, se borrará del esquema.
 **Añadir Conexión** (Edición/Añadir Conexión)	Cambia el modo de edición a "*modo añadir conexión"* (cursor en forma de conexión). Para añadir una conexión habrá que pinchar primero sobre una salida de un OPI y luego, volver a pinchar sobre la entrada de otro OPI.
 **Quitar Conexión** (Edición/Quitar Conexión)	Cambia el modo de edición a *"modo quitar conexión"* (cursor en forma de conexión con borrador). Cada vez que se haga clic sobre una conexión, se borrará del esquema.
 **Ayuda OPI** (Ayuda/Ayuda de los OPIs…)	Cambia el modo de edición a *"modo ayuda"* (cursor en forma de interrogante). Cada vez que se haga clic sobre un OPI, se abrirá una ventana del navegador mostrando el fichero HTML de ayuda correspondiente.
 **Reiniciar OPIs** (Esquema/Reiniciar OPIs)	Reinicia/desactiva todos los OPIs del esquema. De este modo será necesario volverlos a ejecutar para obtener las imágenes de salida.
 **Ejecutar Entradas** (Esquema/Ejecutar Entradas)	Ejecuta automáticamente todos aquellos OPIs del esquema que no tienen entradas.
 **Modo Auto** (Esquema/Auto Ejecución)	Una vez activados los OPIs iniciales del esquema (aquellos que no tienen entradas), el resto se ejecutarán automáticamente de manera consecutiva. Por lo tanto, teniendo activo este modo y pulsando la herramienta anterior *"Ejecutar Entradas"*, podremos ejecutar automáticamente cualquier esquema, independientemente del número de OPIs que contenga.

Tabla 2.1. Opciones de la tabla de herramientas de VISUAL.

2.1.4 Edición y ejecución de esquemas en VISUAL

En este apartado vamos a explicar cómo editar y ejecutar esquemas en VISUAL. Los pasos para crear y editar un esquema nuevo en VISUAL son los siguientes:

1. Crear esquema nuevo: Crearemos un nuevo esquema con la opción *"Nuevo"* de la barra de herramientas. 
2. Añadir OPIs al esquema: Primero seleccionaremos la función deseada en la lista de OPIs de la izquierda y, a continuación, activaremos la herramienta *"Añadir OPI"*  para añadir el OPI seleccionado haciendo clic en el esquema. Repetiremos estas dos operaciones tantas veces como OPIs queramos añadir al esquema.
3. Configurar OPIs: Para configurar los OPIs añadidos, tendremos que establecer las propiedades particulares de cada uno. Para ello, mostraremos el diálogo de propiedades pulsando con el botón derecho del ratón sobre el OPI y seleccionando la opción *"Prop. Particulares"* del menú contextual que aparece.
4. Conectar OPIs: Activaremos la herramienta *"Añadir Conexión"*  e iremos conectando los OPIs: pinchando sobre la salida de un OPI para establecer el principio de una conexión y pinchando sobre la entrada de otro OPI para establecer el fin de dicha conexión.

Después de crear el nuevo esquema, ya podemos ejecutarlo. Para ello, simplemente tendremos que ir pulsando cada uno de los interruptores de los OPIs de manera consecutiva, según el orden marcado por las conexiones. En el recuadro inferior del entorno se mostrarán mensajes sobre el resultado de la ejecución.

El color del pequeño rectángulo situado en la parte superior izquierda de cada OPI nos indicará su estado de ejecución: verde (si no se ha ejecutado todavía o ha sido reiniciado), amarillo (si está ejecutándose), rojo (si ha terminado de ejecutarse correctamente) y azul (si se ha producido algún error durante la ejecución y no ha terminado correctamente). Las conexiones también nos indicarán si los datos fluyen correctamente de un OPI a otro durante la ejecución. Si una conexión aparece cortada, esto significa que el OPI inicio de la conexión todavía no ha creado la imagen de salida que servirá como entrada para el OPI fin de la conexión. Si una conexión aparece continua (sin cortes), esto significa que los datos (imágenes) han pasado de un OPI a otro sin problemas. Debemos destacar que un OPI no se podrá ejecutar hasta que no tenga disponibles todas sus entradas (es decir, hasta que todas las conexiones que llegan a él sean continuas). En la siguiente figura representamos un ejemplo de ejecución de dos OPIs conectados.

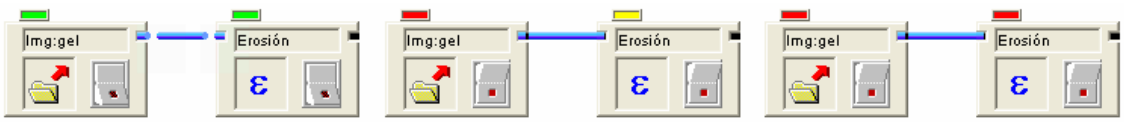

Figura 2.4. Ejemplo de ejecución de OPIs: (a) ningún OPI se ha ejecutado (indicadores en verde y conexión cortada); (b) el primer OPI se ha ejecutado correctamente (indicador en verde y conexión continua al estar disponible la imagen de salida) y el segundo OPI se está ejecutando (indicador en amarillo); (c) los dos OPIs han terminado de ejecutarse correctamente (ambos tienen sus indicadores en rojo).

2.1.5 El visualizador de imágenes de VISUAL

El visualizador de imágenes es una herramienta de VISUAL muy útil para analizar las imágenes de salida de los OPIs que han sido ejecutados. Si después de ejecutar un OPI, acercamos el ratón (en modo de edición normal) a una salida del OPI, veremos como cambia el cursor a una forma de lupa. Si hacemos clic en ese momento, se abrirá el visualizador (figura 2.5) con la imagen de salida seleccionada.

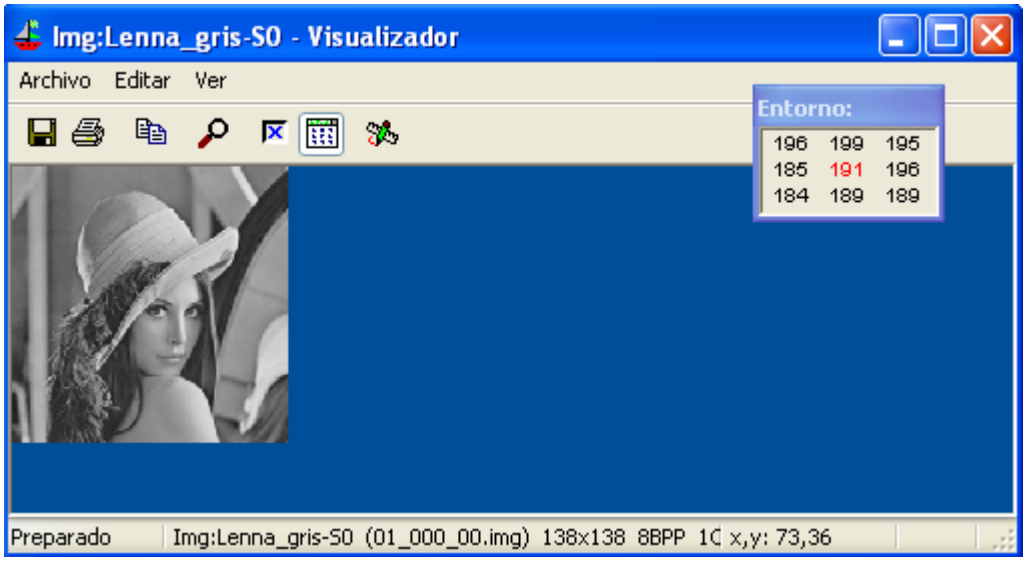

Figura 2.5. Visualizador de imágenes de VISUAL. Se abrirá al hacer clic sobre la salida de un OPI.

Vamos a describir las opciones de la barra de herramientas del visualizador:

Herramienta (Icono/Menú)	Descripción
Guardar (Archivo/Guardar)	Guarda la imagen mostrada como fichero *bmp* en la ruta indicada por el usuario.
Imprimir	Imprime la imagen que está mostrando el visualizador.
Copiar (Edición/Copiar)	Copia la imagen en el Portapapeles de Windows.
Escala (Ver/Escala)	Muestra un pequeño diálogo para indicar la escala de la imagen. Se podrá aumentar y disminuir su tamaño.
Preferencias (Archivo/Preferencias)	Muestra un diálogo con el que podemos configurar las preferencias del visualizador: escala a partir de la que se muestra una rejilla para separar los píxeles, valores de los píxeles numéricos o en porcentajes y tamaño de la ventana de valores.
Ventana Valores (Ver/Ventana de valores)	Muestra una pequeña ventana (*Entorno* en la figura 2.5) donde se representan los valores de los píxeles vecinos del píxel sobre el que está el cursor.
Redibujar (Ver/Redibujar Ventana)	Redibuja la imagen mostrada.

Tabla 2.2. Tabla con las opciones de la barra de herramientas del visualizador de imágenes de VISUAL.

2.2 Guía para el programador de módulos en VISUAL

Con este apartado pretendemos describir el entorno VISUAL desde el punto de vista de un programador que desea añadir nuevos módulos externos (DLL o EXE) a VISUAL para implementar operaciones de Visión Artificial. No sólo vamos a describir el proceso de crear los módulos, sino también los ficheros de configuración necesarios para añadir dichos módulos a VISUAL.

2.2.1 Creación de módulos

Los módulos son ficheros binarios externos (ejecutables EXE o bibliotecas de enlace dinámico DLL) que implementan todas las funciones de un determinado OPI de VISUAL.

Para crear los módulos podremos utilizar cualquier entorno de programación que nos permita desarrollar ficheros EXE o DLLs para MS® Windows. En nuestro caso, hemos utilizado MS® Visual C++ 6.0 para programar los módulos DLL de este proyecto.

Cuando se ejecuta un módulo, éste recibe como entradas un conjunto de parámetros necesarios para su ejecución: las propiedades particulares del módulo (son un conjunto de parámetros propios de cada OPI), los nombres de las imágenes de entrada (imágenes a procesar) y los nombres de las imágenes de salida (imágenes resultado). A continuación, vamos a describir como crear los dos tipos de módulos (EXE y DLL) y como se les pasan los parámetros.

2.2.1.1 Creación de módulos EXE

Los módulos EXE deberán ser compilados como aplicaciones de consola Win32 (*Win32 Console Application*). Un módulo EXE recibirá los parámetros de entrada que necesita para su ejecución como argumentos de la línea de comandos (cadenas de texto tras el nombre de la aplicación) en el siguiente orden:

- Nombre y camino del módulo: Por compatibilidad con el intérprete de comandos, el primer parámetro coincide con la ruta y el nombre completo del módulo. Este parámetro no se utilizará normalmente dentro del código del módulo.
- Función a realizar: Número entero (a partir de 0) que indica la función a realizar dentro del módulo. Debemos recordar que cada módulo (OPI) podrá implementar más de una función.
- Número de entradas: Número de imágenes de entrada que serán tratadas por el módulo.
- Número de salidas: Número de imágenes de salida (resultados) que serán generadas por el módulo.
- Directorio temporal: Directorio temporal donde se almacenarán las imágenes de entrada y salida.
- Datos de configuración de entrada: Lista de parámetros con los valores de los datos de configuración de entrada que requiere el módulo. Cada valor será un parámetro distinto en la línea de comandos. El número de parámetros y su tipo serán especificados por el fichero de interfaz correspondiente. El orden en que

se reciben los datos es el mismo que el orden en el que se definen los parámetros en el archivo de interfaz. Los valores de dichos parámetros serán establecidos por el usuario a través del diálogo de *Propiedades Particulares*, haciendo clic con el botón derecho del ratón sobre el OPI correspondiente.

- Nombres de las imágenes de entrada: Nombres (sin ruta) de las imágenes de entrada que procesará el módulo. Se pondrán tantos parámetros en la línea de comandos como número de imágenes de entrada.
- Nombres de las imágenes de salida: Nombres (sin ruta) de las imágenes que debe generar el módulo. Se pondrán tantos parámetros en la línea de comandos como número de imágenes de salida.

Para que las cadenas de texto de los parámetros se interpreten correctamente, en ellas se han sustituido los caracteres de espacios en blanco por caracteres *asterisco* "*". Por lo tanto, el módulo deberá realizar la conversión inversa para las distintas cadenas pasadas como parámetros a través de las líneas de comandos antes de poder utilizarlas.

Ya que un módulo EXE no es más que un ejecutable, normalmente escrito en C/C++, el número de parámetros (*int argc*) y las cadenas que los representan (*char* argv[]*) se pasarán como parámetros a la función *main* del módulo:

```
int main (int argc, char* argv[])
```

Aquellos parámetros que representen valores numéricos (P. Ej.: tamaños de máscaras, valores de píxeles…) deberán ser convertidos de cadenas de caracteres a variables numéricas mediante alguna función como: *sscanf*, *atof*, *atoi*, *atol*, *strtod*, *strtol*, etc.

No es necesario, en un principio, comprobar los parámetros recibidos. Sin embargo, es recomendable verificar el número de entradas, salidas y propiedades de configuración para evitar excepciones al acceder a posiciones no válidas de los vectores con los nombres de las imágenes o los valores de los parámetros.

2.2.1.2 Creación de módulos DLL

Una librería de enlace dinámico (DLL, *Dynamic-Link Library*) es un archivo ejecutable que actúa como una librería compartida de funciones que pueden ser invocadas simultáneamente por distintos procesos (programas en ejecución). El enlace dinámico proporciona a los procesos una forma de invocar una función sin que ésta forme parte de su código fuente. De este modo, el código fuente de las funciones exportadas por una DLL se compila, se enlaza y se almacena de forma independiente al código fuente de los procesos que la utilizan.

Las DLLs constituyen la evolución de las librerías estáticas y proporcionan un conjunto de ventajas reseñables:

- Reducen el tamaño de los archivos ejecutables: El código de la DLL se almacena en un fichero distinto (fichero con extesión *dll*) al fichero ejecutable que utiliza dicha DLL.
- Pueden estar compartidas entre varias aplicaciones, mejorando el aprovechamiento de la memoria del sistema: Varios procesos pueden usar simultáneamente una DLL, compartiendo una única copia de ella en memoria. De este modo, se reduce la cantidad de memoria necesaria para la ejecución de los procesos. Además, como hay un solo fichero DLL en disco (y no una copia

por cada fichero ejecutable que la utiliza), también se ahorrará espacio en disco.

- Facilitan el mantenimiento del software: Se puede modificar el código de las funciones de una DLL sin tener que recompilar las aplicaciones clientes que la utilizan.

En el caso concreto de VISUAL, los módulos DLL nos proporcionan también una serie de ventajas adicionales respecto a los módulos EXE: se cargan más rápidamente en memoria, no muestran una ventana de consola durante su ejecución y sólo se cargan una vez aunque sean utilizados por muchas operaciones de los esquemas en ejecución. Debido a estas ventajas, utilizaremos módulos DLL para implementar las operaciones de este proyecto. Sin embargo, debemos destacar que la programación y la depuración de módulos DLLs son más complejas que las de módulos EXE.

Una DLL "exporta" una serie de funciones para que un programa cliente las pueda "importar" y ejecutar. En nuestro caso, los módulos DLL que programemos tendrán que definir una función exportable denominada *Ejecutar* que será importada por VISUAL para poder invocar las funciones del OPI correspondiente. La signatura de esta función es la siguiente:

```
int Ejecutar(int iFunc, int iNumCtr, int iNumEnt, int iNumSal, CString& strDirTemp, CString* pStrDatos,
CString* pStrImgEntr, CString* pStrImgSal)
```

Para que esta función sea exportable, deberemos declararla dentro del apartado *EXPORTS* del archivo *def* del proyecto de la DLL. A continuación, mostramos como ejemplo el fichero *def* del módulo *color.dll* (transformaciones entre modelos de color):

```
; color.def : Declares the module parameters for the DLL.

LIBRARY      "color"
DESCRIPTION  'color Windows Dynamic Link Library'

EXPORTS
   ; Explicit exports can go here
   ?Ejecutar@@YAHHHHHAAVCString@@PAV1@11@Z   @1
```

Hemos marcado en negrita la línea que debemos añadir al fichero *def*. Esta línea está compuesta por dos elementos: nombre decorado de la función a exportar ("*?Ejecutar@@YAHHHHHAAVCString@@PAV1@11@Z*") y el ordinal de la función precedido del símbolo "@" (*"@1"*). El nombre decorado de una función (*decorated name*) es la representación interna de la función que utiliza el compilador. Para conocer el nombre decorado de una función tendremos que añadir el atributo */MAP* al linkador (Menú "*Project/Settings/Link/Generate mapfile*" de Visual C++). Con esta opción se generará un fichero de texto con extensión *map* que contendrá una lista con todos los nombres decorados de las funciones utilizadas en la DLL. Afortunadamente, el nombre decorado de la función *Ejecutar* siempre será el indicado en este ejemplo (si utilizamos el compilador de Visual C++ 6.0) y será el mismo para todas las DLLs que vamos a implementar. El ordinal de la función es simplemente un número que indica el índice de la función exportada (ya que en nuestro caso sólo exportaremos una función, el ordinal será siempre el número 1). Este número será utilizado por VISUAL para importar la función.

El significado de los parámetros que recibe el módulo DLL de VISUAL a través de la función *Ejecutar* es el siguiente:

- iFunc: Número de función a realizar (0, 1, 2,…) dentro del OPI.
- iNumCtr: Número de controles o datos de configuración. Recordemos que el número, formato y orden de estos parámetros estarán definidos en el fichero de interfaz del módulo. Los valores de estos parámetros serán establecidos por el usuario a través del diálogo *Propiedades Particulares* del OPI correspondiente.
- iNumEnt: Número de imágenes de entrada que serán tratadas por el módulo.
- iNumSal: Número de imágenes de salida (resultados) que serán devueltas por el módulo.
- strDirTemp: Ruta del directorio donde se almacenarán temporalmente las imágenes de entrada y salida.
- pStrDatos: Vector con los datos de configuración, todos como cadenas de caracteres.
- pStrImgEntr: Vector con los nombres (sin ruta) de las imágenes de entrada.
- pStrImgSal: Vector con los nombres (sin ruta) de las imágenes de salida.

Al igual que en los módulos EXE, no es necesario comprobar los parámetros recibidos. No obstante, es recomendable comprobar el número de elementos para evitar el acceso a posiciones no válidas de los vectores con los nombres de las imágenes o con los valores de los parámetros.

En la lista de parámetros de la función *Ejecutar*, podemos observar que algunos de los parámetros son objetos de la clase *CString*. Esta clase pertenece a la librería MFC (*MS® Foundation Classes*), que facilita la programación en Windows. Por ello, en todos los ficheros de código de nuestras DLLs tendremos que incluir esta librería (#include <stdafx.h>). Además, los módulos DLLs que creemos deberán ser *DLLs MFC* (DLLs que usan internamente la librería MFC). Existen dos tipos de DLLs MFC que podremos utilizar:

- DLLs enlazadas estáticamente a la librería MFC (*Regular DLL using MFC statically linked*): Estas DLLs incluirán una copia de todo el código de la librería MFC. Cada DLL tendrá su propia copia de la librería.
- DLLs enlazadas dinámicamente a la librería MFC (*Regular DLL using shared MFC DLL*): Estas DLLs no incluirán el código de la biblioteca MFC sino que importarán dinámicamente las clases/funciones necesarias de los ficheros DLL de la librería MFC (ficheros *mfcXX.dll* -siendo *XX* la versión- y *msvcrt.dll*).

El enlace dinámico nos permitirá obtener DLLs con un tamaño sustancialmente menor (pasando de DLLs de unos 800Kb con enlace estático a DLLs de unos 40Kb con enlace dinámico). Sin embargo, si utilizamos el enlace dinámico, debemos asegurarnos de que las DLLs de MFC (ficheros *mfcXX.dll* y *msvcrt.dll*) estén disponibles. Afortudamente, en las últimas versiones de MS® Windows estas librerías vienen de serie con el sistema operativo (en la carpeta *C:\windows\system32*). Por ello, todos los módulos DLLs que hemos desarrollado en este proyecto están enlazados dinámicamente con la librería MFC.

2.2.2 Imágenes de entrada y salida

Las imágenes de entrada a operar y las de salida están almacenadas en el directorio temporal de intercambio (que seleccionamos en el proceso de *configuración de directorios* durante la instalación de VISUAL). Son imágenes sin comprimir que admiten varios planos o canales de dos dimensiones. También permiten especificar el tamaño de cada valor almacenado en cada plano.

El formato de un archivo de imagen BMP obedece a la siguiente estructura (un WORD son dos BYTEs):

```
WORD wXSize
WORD wYSize
WORD wChannels
WORD wPixelSize
BYTE data [wXSize * wYSize * wChannels * wPixelSize]
```

Los primeros cuatro elementos constituyen la cabecera del fichero imagen y representan respectivamente los siguientes datos: el tamaño a lo largo del eje X de la imagen (ancho o número de columnas de la imagen), el tamaño a lo largo del eje Y de la imagen (alto o número de filas de la imagen), el número de canales o planos de la imagen (las imágenes en escala de grises tendrán un plano y las imágenes en color tendrán tres planos, RGB) y el número de bytes que ocupa un valor de un plano (es decir, el tamaño de cada componente/canal de un píxel).

Después de la cabecera se encuentra un vector con los datos de la imagen sin comprimir. Los datos de la imagen se ordenan alternando los valores de los canales para cada píxel por cada fila de la imagen, empezando desde la fila superior. Por ejemplo, para una imagen en color RGB, obtendríamos la siguiente secuencia de datos:

	Columna 1	**Columna 2**	**…**	**Columna X**
Fila 1	(R11,G11,B11)	(R12,G12,B12)	…	(R1X,G1X,B1X)
Fila 2	(R21,G21,B21)	(R22,G22,B22)	…	(R2X,G2X,B2X)
...	…	…	…	…
Fila Y	(RY1,GY1,BY1)	(RY2,GY2,BY2)	…	(RYX,GYX,BYX)

Tabla 2.3. Ordenación de los datos de una imagen en color RGB.

Tal como hemos indicado, el valor de un canal (o componente) de un píxel puede ocupar uno o más bytes, siendo lo habitual trabajar con componentes de un byte (valores en el rango [0,255]). Así, por ejemplo, la componente R (Roja) de un píxel de una imagen en color ocupará un byte y, por lo tanto, el píxel completo (formado por las tres componentes: R, G y B) ocupará tres bytes.

No obstante, también es posible trabajar con imágenes con otros tamaños de píxel. En la tabla 2.4 mostramos todos los tamaños posibles para un canal de un píxel. Los valores en coma flotante pueden ocupar demasiada memoria, pero son muy útiles para devolver resultados numéricos o estadísticos. Las imágenes con píxeles en coma flotante suelen usarse como datos de entrada para OPIs de visualización de resultados: gráficas 2D y tablas de datos. Éste es el modo mediante el que VISUAL puede mostrar cálculos numéricos con decimales, que serían imposibles de almacenar en imágenes con píxeles de un solo byte. Por lo tanto, las imágenes con píxeles en coma flotante son realmente contenedores de datos y no son imágenes digitales reales. Para obtener más información sobre el tratamiento de este tipo de imágenes en VISUAL, se puede consultar [CANDELAS03a].

Tipo	Tamaño	Descripción
BYTE	1 byte = 8 bits	Entero sin signo en [0,255]
WORD	2 bytes = 16 bits	Entero sin signo de 16 bits
DWORD	4 bytes = 32 bits	Entero sin signo de 32 bits
int	4 bytes = 32 bits	Entero con signo de 32 bits
float	4 bytes = 32 bits	Coma flotante IEEE con bit de signo, 8 bits de exponente y 23 bits de mantisa
double	8 bytes = 64 bits	Coma flotante IEEE con bit de signo, 11 bits de exponente y 52 bits de mantisa

Tabla 2.4. Tamaños posibles de cada canal de un píxel para imágenes en VISUAL.

2.2.3 Archivo de descripción de módulos

La lista de módulos que la aplicación debe cargar, la jerarquía de clases de éstos (es decir, su organización jerárquica) y sus datos principales se definen en el *archivo de descripción de módulos* (normalmente se llama *modulos.ini*). Este archivo es un fichero de texto cuyo nombre y ruta deberán ser escogidos por el usuario en el menú "*Archivo/Configuración de Directorios"* de VISUAL (tal como explicamos en el apartado 2.1.1).

El archivo se compone de dos bloques de definiciones: el bloque que describe la lista de módulos de los OPIs (con sus funciones correspondientes) y el bloque que define la jerarquía o árbol de clases que se mostrará en la lista de OPIs del entorno gráfico VISUAL. Estos dos bloques se deben delimitar con las siguientes etiquetas:

```
[MODULOS]
   # Definición de módulos y funciones
[END]
[CLASES]
   # Definición de clases (jerarquía de los módulos)
[END]
```

Es indiferente el orden de definición de los dos bloques y su texto se puede escribir en mayúsculas o minúsculas. Se pueden añadir al fichero los siguientes elementos para hacerlo más legible: líneas de comentario (empezarán por el carácter "#"), líneas en blanco, espacios y tabuladores a la izquierda.

VISUAL comprueba este archivo de configuración cada vez que arranca y, en caso de detectar algún error, informará sobre el tipo de error y la línea en donde se encuentra.

Tendremos que editar este archivo cada vez que queramos añadir un nuevo módulo al entorno VISUAL.

En los siguientes apartados vamos a describir cómo se definen los elementos de cada uno de los dos bloques que forman este archivo: definición de módulos y definición de clases.

2.2.3.1 Definición de módulos

Dentro del bloque de módulos se debe describir las características básicas de cada módulo seguidas de las características de sus funciones, del siguiente modo:

```
Nombre_del_OPI Ident_OPI Num_funciones Nombre_módulo [Ayuda]
Nombre_función_1 Id_f_1 Ent_Min_1 Ent_Max_1 Sal_Min_1 Sal_Max_1 Fich_icono_1
Descripción_de_la_función_2
Nombre_función_2 Id_f_2 Ent_Min_2 Ent_Max_2 Sal_Min_2 Sal_Max_2 Fich_icono_2
Descripción_de_la_función_2
```

A continuación se describe cada uno de los parámetros anteriores:

- Nombre_del_OPI: Cadena de texto con el nombre del OPI que se mostrará en la lista de OPIs del entorno VISUAL.
- Ident_OPI: Valor numérico único que identificará el OPI del resto de OPIs.
- Num_funciones: Número de funciones que incluye el OPI, y que se definen a continuación.
- Nombre_módulo: Cadena de texto con el nombre del archivo ejecutable (EXE o DLL) del módulo. No contiene ruta ni extensión (ya que las añade automáticamente el entorno VISUAL). La ruta se corresponderá con el directorio de módulos (ver apartado 2.1.1).
- Ayuda: Nombre de la carpeta que contiene los archivos de ayuda del OPI y que está situada dentro del directorio de archivos de ayuda (ver apartado 2.1.1). Este parámetro es opcional.
- Nombre_función_x: Cadena de texto con el nombre de la función que se mostrará en la lista de OPIs del entorno VISUAL.
- Id_f_x: Valor numérico único dentro del módulo que identificará a la función.
- Ent_Min_x, Ent_Max_x: Número de entradas mínimo y máximo que puede tener la función. Si son iguales, indica que se requiere un número concreto de entradas.
- Sal_Min_x, Sal_Max_x: Número de salidas mínimo y máximo que puede tener la función. Si son iguales, indica que ser requiere un número concreto de salidas.
- Fich_icono_x: Nombre del fichero BMP que contiene el icono de la función. No se incluye la ruta ya que se corresponde con el directorio de imágenes (ver apartado 2.1.1), ni la extensión (que siempre será *bmp*).
- Descripción_de_la_función_x: Frase con la que se describirá la funcionalidad de la función correspondiente. Esta descripción se mostrará en el recuadro de *Información de OPIs* del entorno gráfico VISUAL cuando la función haya sido seleccionada en la lista de OPIs.

Las cadenas de caracteres con nombres no pueden contener espacios (excepto para las líneas de descripción de funciones), aunque el carácter "_" será reemplazado por un espacio en la interfaz. Los valores numéricos deben ser siempre enteros no negativos.

Los valores de identificador de OPI para los distintos módulos definidos deben ser distintos. También deben ser distintos los valores de identificador de función dentro del grupo de funciones de un módulo. Es importante no cambiar estos valores en un fichero

de configuración que ya ha sido utilizado por el entorno VISUAL para dibujar esquemas, ya que el esquema podría malinterpretarse. Además, los identificadores de OPIs y funciones deben ser valores comprendidos entre 0 (incluido) y 30000 (no incluido), ya que los valores de 30000 o mayores son usados por el entorno VISUAL para sus OPIs internos.

2.2.3.2 Definición de clases

Dentro del bloque de clases se define un árbol de jerarquía de tipos de OPIs. Este árbol se especifica mediante sentencias como estas:

```
Clase_Padre_1 Número_Clases_Hijas_1
        Clase_Hija_11
        Clase_Hija_12
        Clase_Hija_13
Clase_Padre_2   Número_Clases_Hijas_2
        Clase_Hija_21
        Clase_Hija_22
        Clase_Hija_23
```

Una *Clase_Padre* es una nueva clase de OPIs que engloba varios de éstos. Una *Clase_Hija* es un nombre de OPI o una Clase_Padre definida anteriormente. De este modo, el árbol se define desde las ramas hacia la raíz. La raíz del árbol se indica con el carácter "*" como valor de nueva Clase_Padre.

2.2.3.3 Ejemplo de archivo de descripción de módulos

A continuación, mostramos un ejemplo de un archivo de descripción de módulos. A la derecha hemos añadido una figura con el árbol de OPIs que aparecerá en el entorno gráfico VISUAL al utilizar este archivo de descripción de módulos.

```
[MODULOS]

Constante 1 1 constante
  generar 0 0 0 1 1 cons
  Genera una imagen constante

Aritméticas 2 4 aritmeticas SIArit
  Suma 0 2 2 1 1 oa_sum
  Suma dos imagenes
  Resta 1 2 2 1 1 oa_res
  Resta dos imagenes
  Producto 2 2 2 1 1 oa_prod
  Multiplica dos imagenes
  División 3 2 2 1 1 oa_div
  Divide dos imagenes

RGB 3 1 rgb
  SepararRGB 0 1 1 3 3 rgb_sep
  Separa los canales de una imagen RGB

HSI 4 2 hsi
  SepararHSI 0 1 1 3 3 hsi_sep
  Transforma una imagen RGB en HSI/LAB
  JuntarHSI 1 3 3 1 1 hsi_jun
  Transforma los canales HSI/LAB en una imagen RGB

Compara 5 1 cmpImg SIOtr
  Compara 0 2 2 0 0 cmpImg
  Compara pixel a pixel dos imagenes

[FIN]
```

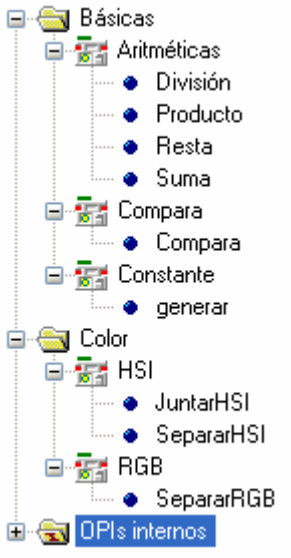

Figura 2.6. Lista de OPIs en VISUAL.

```
[CLASES]

Básicas 3
  Constante
  Compara
  Aritméticas

Color 2
  RGB
  HSI

* 2
  Básicas
  Color

[FIN]
```

2.2.4 Archivos de interfaz

Estos ficheros definen el aspecto del diálogo *"Propiedades Particulares"* que VISUAL presenta al usuario cuando éste solicita cambiar los datos de configuración de entrada de un OPI (haciendo clic con el botón derecho del ratón sobre el OPI correspondiente en el esquema).

Cada módulo tiene su propio fichero de interfaz, común para todas sus funciones. Este fichero tendrá el mismo nombre que el módulo correspondiente, pero con extensión *ini* y se almacenará en el directorio de ficheros de interfaz (ver apartado 2.1.1). En este fichero se define el número y el tipo de datos de configuración requeridos por el módulo (junto con el tipo de control de Windows utilizado para que el usuario introduzca el valor en la interfaz). Su formato es el siguiente:

```
Nombre_par_1 Tipo_Control_1 x_1 y_1 ancho_1 alto_1 Valor_min_1 Valor_max_1 Valor_defecto_1
Nombre_par_2 Tipo_Control_2 x_2 y_2 ancho_2 alto_2 Valor_min_2 Valor_max_2 Valor_defecto_2
…
```

Debe existir una línea para cada control en donde se especifican: su nombre, su tipo (según lo indicado en la tabla 2.5), la posición (x, y) del punto superior izquierdo de su rectángulo dentro del diálogo, sus dimensiones (ancho y alto), sus valores mínimo y máximo y valor por defecto. El fichero admite además líneas de comentario tras el símbolo "#", así como líneas en blanco.

Aunque siempre se interpretan los valores de posición *x, y,* y *ancho*, el valor de *alto* depende del tipo de control. Por ejemplo, los controles de edición numérica y de archivo mantienen un alto fijo. Un control de *selección* usa el valor de *alto* para definir la separación entre opciones. Sólo los controles de cadena de texto y texto estático usan el valor de *alto* como tamaño de altura.

Los valores máximo y mínimo se aplican a los controles numéricos. En el caso de cadenas de texto se utiliza también el valor máximo para indicar su longitud máxima. Para el control de selección, el valor máximo indica el número de opciones.

Excepto las cadenas de texto estático, el resto de valores definidos en un fichero de interfaz serán recibidos por el módulo correspondiente como cadenas de texto en el orden definido.

El área disponible para dibujar controles es un rectángulo de 380 píxeles de ancho por 190 píxeles de alto. Aunque en la carga no se verifican las coordenadas de los controles y sus tamaños, éstos deben estar incluidos en este rectángulo para que sean creados.

Los posibles tipos definidos actualmente, así como los parámetros y controles asociados se muestran en la siguiente figura:

Tipo	Denominación	Tipo parámetro	Control Windows	Descripción
0	Texto estático	-	Static	Sólo para aspecto gráfico del diálogo
1	Valor booleano	>0 TRUE ≤0 FALSE	Check Box	Valor booleano
2	Cadena de texto	char[]	Edit	Para definir cadenas de texto, que pueden tener varias líneas
3	Número real	double	Edit	Para definir números reales
4	Número entero	long	Edit	Para definir número enteros
5	Selección	int (positivo)	Radio Button	Para definir opciones de selección
6	Archivo a abrir	char[]	Edit + File Dialog	Permite que el usuario especifique un archivo existente para abrir
7	Archivo a guardar	char[]	Edit + File Dialog	Permite que el usuario especifique un archivo para escribir
8	Control Slide	long	Slide	Selección de un valor entero con una barra horizontal con etiquetas
9	Entero con Spin	long	Edit + Spin	Selección de un valor entero con un control de incremento-decremento

Tabla 2.5. Tipos de controles definidos para un fichero de interfaz en VISUAL.

Un ejemplo de archivo de interfaz puede ser el siguiente (más abajo mostramos el cuadro de diálogo resultante):

```
Etiqueta    0   0   0   200  22   0     0    0
PReal       3   0   25  200  22   0     100  1
PEntero     4   0   50  200  22   -100  100  1
PArchivo    6   0   75  200  22   0     0    imagen.bmp
PSlider     8   0   100 200  18   0     30   15
PSelección  5   220 0   200  18   0     3    0
PSpin       9   220 100 50   18   0     300  150
```

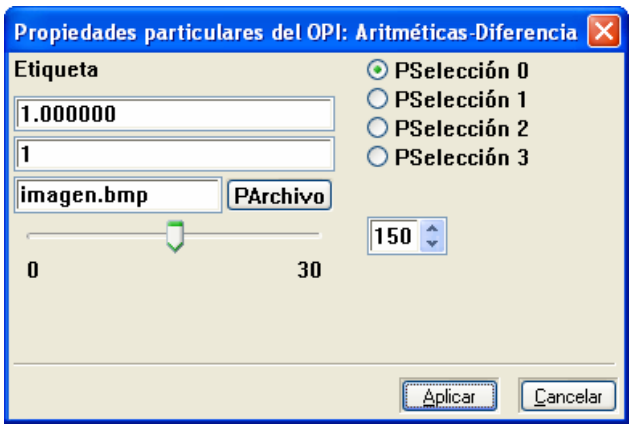

Figura 2.7. Cuadro de diálogo de ejemplo de las propiedades particulares de un OPI.

2.2.5 Códigos de error

Un módulo puede devolver códigos de error a través del valor de retorno de su función principal. En el caso de los módulos EXE, el código de error es el valor entero devuelto por la función *main*. En el caso de los módulos DLL, el código de error es el valor entero devuelto por la función *Ejecutar*.

El valor 0 será interpretado como ausencia de errores (generación de las imágenes resultado sin problemas). Los valores positivos entre 0 y 100 indican errores en la ejecución (no se han generado correctamente las imágenes resultado). Los valores superiores a 100 están reservados para códigos internos de VISUAL.

Cada OPI puede tener asociado un archivo de mensajes de error, donde se especifican las cadenas de texto correspondientes a los códigos de error de usuario devueltos por un OPI (de 1 a 100). El archivo de mensajes de un OPI debe tener el mismo nombre que el módulo y el archivo de interfaz del OPI, pero con la extensión *txt*, y debe estar en la carpeta de interfaces. Si no se especifica dicho archivo, simplemente se muestra el número del error en el entorno. Si se crea ese archivo y se definen los mensajes, el texto del mensaje correspondiente se muestra en el entorno VISUAL junto al número de error. El archivo debe ser de texto llano, con la siguiente estructura:

- Las líneas que comienzan con el carácter "#" se ignoran. Servirán como comentarios. También se pueden añadir líneas en blanco para facilitar la legibilidad del archivo.
- Cada mensaje se define en una línea acabada con un <CR><LF>.
- La línea puede comenzar con espacios o tabuladores, después debe tener el número del código de error, sigue una coma, y el resto hasta el final de línea se interpreta como una cadena de texto que representa el mensaje.

A continuación, mostramos un pequeño ejemplo de un fichero con mensajes de error:

```
#Principio del fichero de errores
71,Error al intentar abrir el fichero de la imagen fuente.
72,Error al leer la cabecera de la imagen.
73,El formato de la imagen no es el correcto.
74,Error al leer los píxeles de la imagen.
#Fin del fichero de errors
```

2.2.6 Sistema de ayuda

Cada OPI puede tener asociado un conjunto de archivos de ayuda en formato HTML. Los ficheros de ayuda de un determinado OPI se guardarán en un subdirectorio del *directorio de archivos de ayuda* (ver apartado 2.1.1). El nombre de dicho subdirectorio debe coincidir con el especificado en la definición del módulo dentro del *archivo de definición de módulos* (ver apartado 2.2.3). Cada módulo podrá tener su propia carpeta con sus archivos de ayuda.

La ayuda de un módulo es opcional y en el *archivo de definición de módulos* se puede obviar la indicación de la carpeta de ayuda. En tal caso, el módulo correspondiente no tendrá ningún fichero de ayuda.

Dentro de la carpeta de ayuda de un módulo, el único archivo obligatorio es *index.html*. Éste es el archivo que abrirá el entorno VISUAL en una ventana del navegador para mostrar la ayuda de un OPI. Se podrán añadir más archivos de ayuda que deberán ser accesibles a partir del archivo anterior mediante hipervínculos.

Para visualizar la ayuda desde VISUAL, primero se debe seleccionar el navegador que utilizará la aplicación para mostrar la ayuda. Esto se hace mediante el diálogo al que se accede con la opción del menú *"Archivo/Configurar Aplicación…"*. Se puede seleccionar entre usar el navegador MS® Internet Explorer u otro (Firefox, Netscape, Opera…).

Para visualizar la ayuda de un OPI existen tres alternativas:

- En la lista de OPIs del entorno VISUAL, cuando se selecciona un OPI que dispone de ayuda, se activa el botón "*Ayuda"* (que se encuentra justo debajo de la lista). Si se pulsa ese botón, se abrirá el navegador para mostrar la ayuda del OPI.
- Al pulsar con el botón derecho del ratón sobre un OPI de un esquema, se puede seleccionar la opción *"Ayuda sobre el OPI…"* del menú contextual.
- Se puede activar el modo de edición *"ayuda"* (cursor en forma de interrogante) mediante la opción *"Ayuda OPI"*  de la barra de herramientas o mediante el menú *"Ayuda/Ayuda de los OPIs…"*. Mientras que esté activo este modo, cada vez que hagamos clic sobre un OPI del esquema, se mostrará su ayuda (en caso de que esté disponible).

Si se desea obtener más ayuda sobre VISUAL, se puede consultar el manual [CANDELAS03b]. Si se desea descargar la última versión de VISUAL, se puede consultar la web [VISUAL04]. Para obtener más información sobre la programación de DLLs y la utilización del entorno MS® Visual C++, se pueden consultar [KRUGLINSKI96] y [ZARATIAN99].