UNIVERSIDAD POLITECNICA DE VALENCIA

Configuración de entornos de desarrollo para la creación de aplicaciones utilizando Visión por Computador OpenCV

Apellidos, nombre	Agustí Melchor, Manuel[1] (magusti@disca.upv.es)
Departamento	[1]Dpto. De Ing. De Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSIDAD POLITECNICA DE VALENCIA

1 Resumen

En este artículo vamos a presentar una introducción al desarrollo de aplicaciones en lenguaje C que utilizan funciones documentadas de un API, en concreto OpenCV (véase [1], [4] y [2]) . Aunque es un material de apoyo, está pensando de forma que la realización práctica de lo presentado en este documento conduzca a la obtención de una serie de resultados experimentales: tanto código como ficheros de imágenes de resultados.

Se va a presentar un enfoque que permita trabajar en cualquier sistema operativo desarrollando sistemas de características multimedia que hagan uso de la imagen, en este caso, como un medio característico del que extraer y sobre el que mostrar información.

2 Introducción

El presente desarrollo está encaminado a ofrecer una base para la creación de proyectos software de mediana envergadura caracterizados por utilizar una librería ya existente y desarrollada por otras personas.

En estos casos se tiene acceso a una versión para desarrollador que pueden enlazar contra las librerías el código que desarrolla y, con un poco de suerte, de la adecuada documentación al respecto.

Existen una serie de recursos en la web de OpenCV[1] sobre diferentes entornos: Visual C++. Eclipse, Builder, DevCpp, así como alternativas sobre GNU/Linux y sobre Mac OS/X.

En este sentido, proponemos aquí el uso de Code::Blocks como entorno de desarrollo multiplataforma, esto es, nos permitirá cambiar de sistema operativo de soporte, pero seguir los desarrollos en un entorno común y conocido. Lo cual siempre es una ventaja a tener en cuenta.

3 Objetivos

Los objetivos de esta práctica son:

- Dar a conocer al alumno, de un modo participativo, el uso de las funciones de OpenCV dentro del código que uno pueda desarrollar.
- Utilizar una plataforma de experimentación abierta y multiplataforma que permita al alumno proseguir su autoaprendizaje de forma independiente.

Una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Compilar los programas que utilizan la librería OpenCV en diferentes sistemas operativos.
- Generar un ejecutable a partir del código fuente que utilice las librerías OpenCV.

[1]Véase en <http://opencv.willowgarage.com/wiki/FullOpenCVWiki#Welcome.2BAC8-Introduction.GettingStartedwithOpenCV<.

- Determinar la configuración adecuada en función de dónde estén instaladas las librerías y soportes de ejecución de OpenCV.
- Configurar y utilizar entornos de desarrollo visuales como *Code::Blocks* o *Visual Studio* 2005/2008.

4 Desarrollo

En la actualidad los computadores tiene potencia de cálculo suficiente como manipular tipos de datos complejos, en particular, al disponer de periféricos que permitan mostrar y adquirir información en formato visual, las capacidades de comunicación entre las personas y las máquinas han alcanzado un punto interesante al abrir un camino "natural" de interacción entre ambos.

Antes de iniciar una tarea más compleja es necesario conocer las bases, así que es momento de ponernos manos a la obra. Nuestro programa va a ser muy simple, puesto que el interés es compilarlo y obtener un ejecutable independiente del entorno de desarrollo.

En nuestro caso el programa a crear lo tomaremos de los ejemplos que acompañan a OpenCV: *drawing.c*. ¿Qué qué hace? Lo vemos en un momento. De momento, hagamos una copia a un directorio donde poder trastear con él, en mi caso está instalado en

/usr/share/doc/opencv-doc/examples/c/drawing.c

4.1 Compilando desde la línea de órdenes

En GNU/Linux, para no depender de la instalación de un equipo o de una distribución en concreto, podemos recurrir, si está instalada, a la orden. Esta nos permite averiguar una serie de parámetros de la instalación de unas librerías de desarrollo, como es el caso de OpenCV.

Por ejemplo, podemos averiguar la versión instalada de OpenCV con

```
$ pkg-config opencv –modversion
```

Observa que el símbolo *$* expresa que hay que teclear lo que le sigue en un terminal. La orden es *pkg-config* y recibe dos parámetros:

- El primero, para nosotros, siempre es opencv, por que es la librería de la que queremos saber cosas.
- El segundo es lo que queremos saber, en este caso con --modversion la versión instalada,

Hazlo ahora en un terminal y averigua qué versión tienes instalada. Si no la tienes instalada, poco vamos a poder jugar … Lanza el programa de instalación de paquetes que uses habitualmente (*synaptic*, *aptitude*, *dselect*, ...) o en la opción ya habitual de "Instalar/elimiinar software" de la aplicación de panel de control. Busca todos los paquetes que tengan en su nombre la subcadena "opencv", si aparecen dependencias también las instalaremos.

Debes tener un paquete *opencv-doc* de documentación y ejemplos ya escritos, *libcv*, *libhighgui* y *libcvaux* (las librerías para enlazar, quizá con algún número de versión añadido al nombre) y las versiones de desarrollo de estas (los ficheros de cabecera) con nombres *libcv-dev*, *libhighgui* -dev y *libcvaux-dev* . Las sufijos "-dev" es por "development".

En un fichero *Makefile* o en un *script* en función de si la versión de OpenCV es una u otra realizar accesiones diferentes y en ese caso podemos ver si es una versión en concreto, por ejemplo "¿La versión instalada de OpenCV es la 1.0? Atención a las comillas simples que se usan:

```
$ pkg-config --exists 'opencv == 1.0'
```

O también para el caso de la versión instalada sea posterior a una dada, como por ejemplo "¿Es la versión instalada de OpenCV la 1.0 o una posterior?

```
$ pkg-config --exists 'opencv >= 1.0'
```

Dejamos esto a un lado por que hay que explicar cómo se recofe el resultado de ejecutar esas órdenes y no es ahora el momento. En nuestro caso, nos importa sobre todo el saber qué dependencias se utilizan para compilar un programa en código fuente en lenguaje C contra esta librería de desarrollo. Esto se averigua con

```
$ pkg-config opencv --cflags
```

Y ¿contra qué librerías hay que enlazarlo?

```
$ pkg-config opencv --libs
```

Así para compilar:

```
$ gcc `pkg-config opencv --cflags --libs` drawing.c -o drawing
```

Atención a las comillas invertidas que se han utilizado: las de ejecución. Esto ha hecho que se ejecute la orden *pkg-config* con los parámetros que se le pasan y lo que devuelve esta orden se pasa como parámetros a la orden *gcc*.

Lo que permite crear un ejecutable de nombre *drawing* a partir del fichero fuente *drawing.c*, con el compilador *gcc* y el uso de los ficheros de cabecera y las librerías que son aplicables a los programas que utilizan OpenCV. Que siempre se puede reescribir a la usanza tradicional con su equivalente orden explícita:

```
$ gcc -I/usr/include/opencv -lcxcore -lcv -lhighgui -lcvaux -lml drawing.c -o drawing
```

Que es un poco más "pesada" de escribir ¿verdad? Se ejecuta con la orden

```
$ ./drawing
```

La ejecución termina al pulsar cualquier tecla, la ventana se cerrará y mientras se habrán visto varios ejemplos de primitivas de dibujo en la ventana de la aplicación.

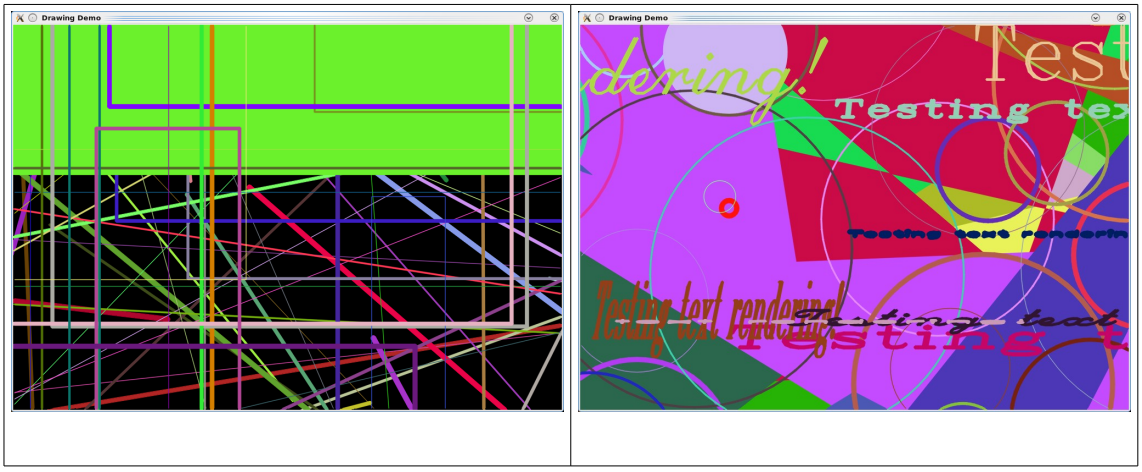

Figura 1: Capturas de dos instantes de la ejecución del programa "drawing".

4.2 El entorno de desarrollo Code:Blocks

Se propone el uso de Code::Blocks [3], que en su versión 8.02 se ha utilizado aquí, como herramienta visual de desarrollo de código, permitiendo su escritura, verificación, depuración y configuración del ejecutable creado.

En el Wiki de OpenCV [4] se puede ver un resumen similar para trabajar con Code:.Blocks en *MS/Windows*, se observará que es análogo a cómo crear un proyecto en este entorno de desarrollo en la plataforma *GNU/Linux* basado en la distribución Ubuntu que se muestra a continuación.

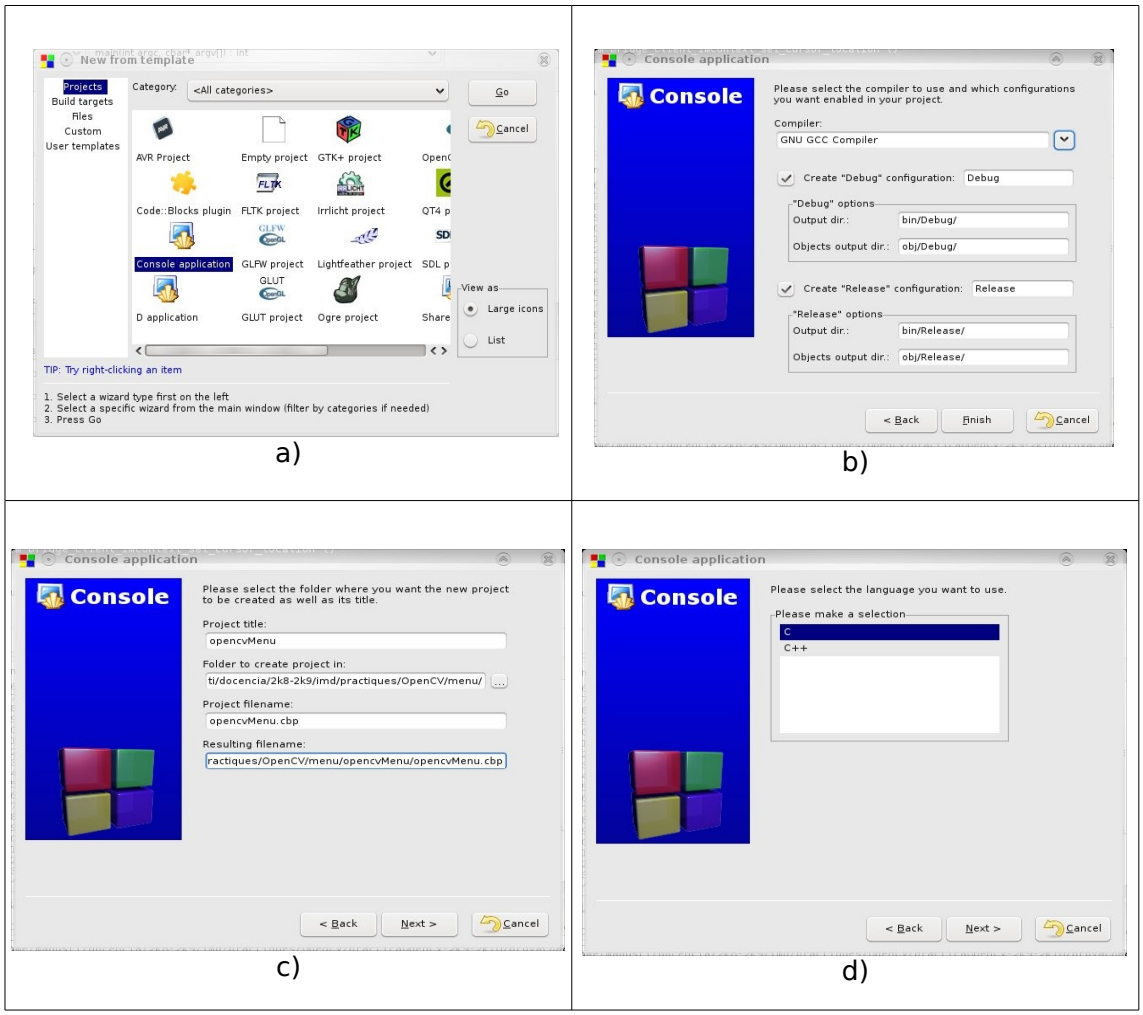

Figura 2: Creación de un proyecto: primeros pasos.

Se puede empezar creando un proyecto desde cero con la opción de menú *File | New | Project*, con lo que aparece la caja de diálogo *New from template* (Figura 2a), donde se habrá de escoger el tipo de proyecto (*console application*) y aceptar con el botón "*Go*". A partir de aquí el asistente nos guiará por las opciones de cada uno, en nuestro caso escogeremos el compilador (Figura 2b), el directorio donde se ubicarán los ficheros, configuraciones para las versiones de

desarrollo y de entrega, el nombre del proyecto (Figura 2c) y el lenguaje de programación (Figura 2d).

Una vez abierto, el entorno ofrecerá una apariencia similar a la mostrada en la Figura 3.que muestra un momento de la etapa de depuración de un ejemplo de creación de imágenes.

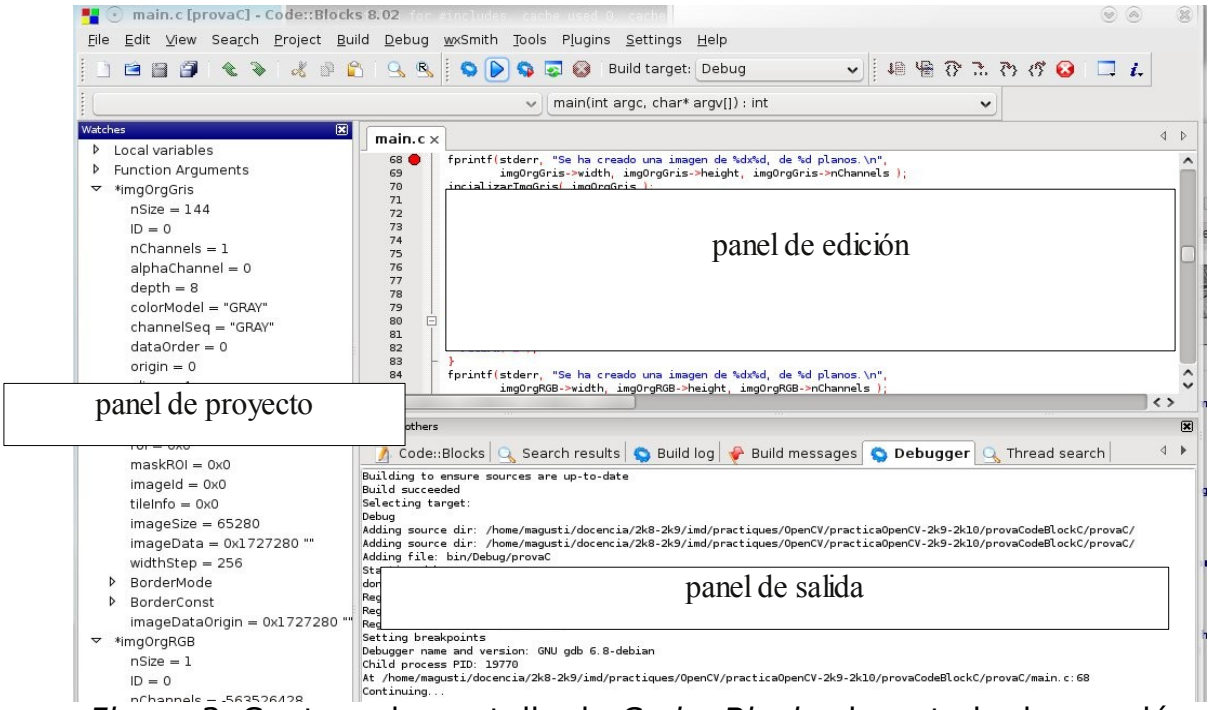

Figura 3: Captura de pantalla de Code::Blocks durante la depuración del primer ejemplo de la práctica.

Siempre se puede encontrar esta información para actualizarla en la sección de *Management* (el panel de proyecto) dentro de la pestaña *Projects*, el área de trabajo (*Workspace*) del proyecto en que estamos interesados y al pulsar con el botón secundario en el nombre del proyecto.

Es interesante establecer algunas opciones más como que se genere información para poder depurar el proyecto (Figura 4a), para ello hay que activar la opción correspondiente en el menú contextual del proyecto: apartado *Build Options...* Para poder utilizar las librerías de OpenCV se habrá de indicar en el menú contextual del proyecto: en su apartado *Properties... > Libraries* (Figura 4b-d)hay que escoger *Known libraries* "opencv" y utilizando el botón "<", que está en el centro de las cajas de texto, aparecerá en *Libraries used in project*. Estas etapas se muestran en las cuatro siguientes imágenes.

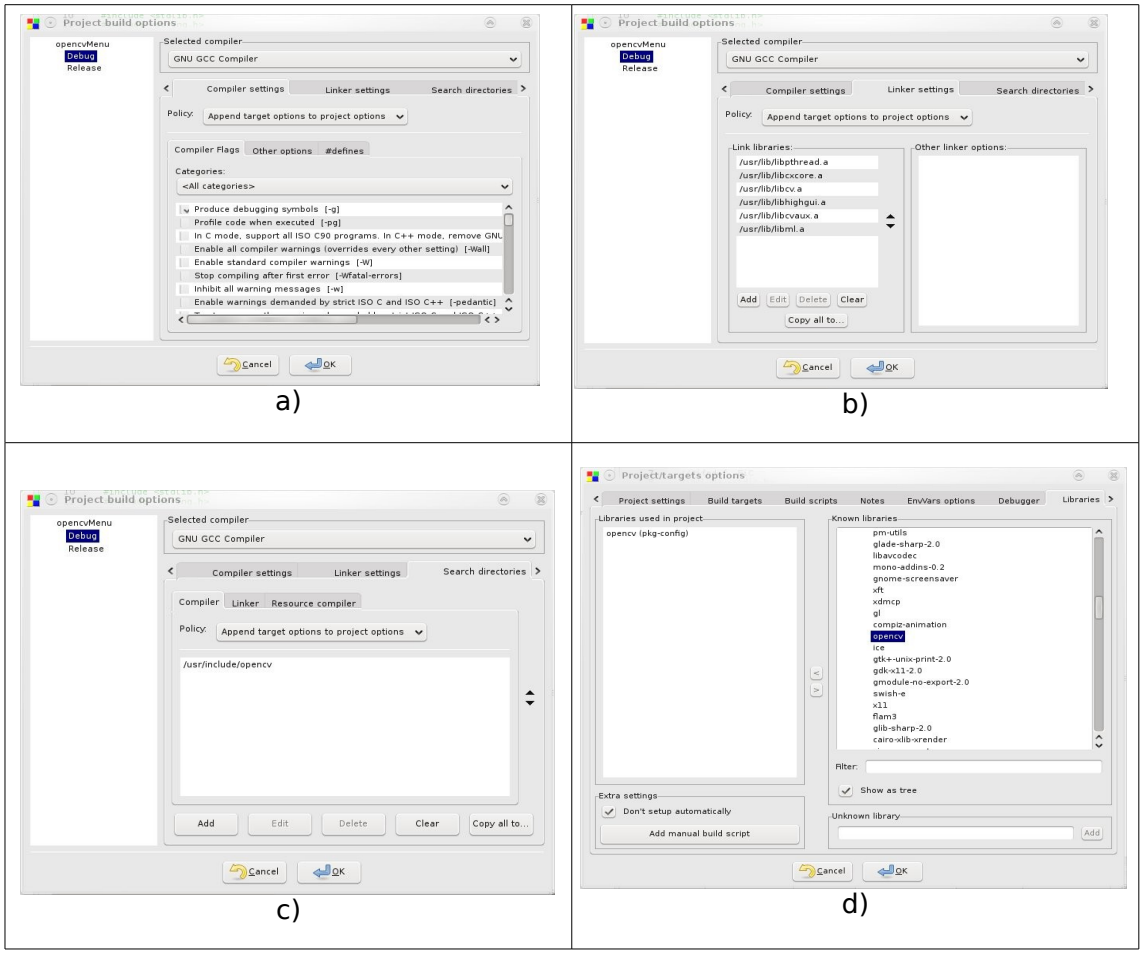

Figura 4: Estableciendo la configuración para el ejecutable final.

4.3 Organización en paneles

Como se mostraba en la Figura 3 el entorno se divide en tres paneles: *Management*, edición y *Logs & others*. Que se describen a continuación.

El **panel de proyecto** (*Management*) permite seleccionar tres vistas diferentes del proyecto:

1. ***Projects*** Permite acceder a los archivos que componen el proyecto.
2. ***Resources*** Permite modificar los recursos de la aplicación (menús, iconos, cuadros de diálogo, etc.)
3. ***Symbols.*** Muestra la jerarquía de clases, así como los métodos y atributos asociados a cada una.

En el **panel de edición** se realiza la edición del código fuente, los ficheros de cabeceras y de los recursos del proyecto.

En el **panel de salida** se obtienen los informes generados por el proceso de compilación y depuración y los resultados de las búsquedas realizadas.

4.4 El Menú de compilación *(Build) y depuración (Debug)*

Las opciones más interesantes para el desarrollo de esta práctica se encuentran centradas en las opciones de menú Build y Debug, estas se muestras desplegadas para familiarizarse con los atajos de teclado (Figura 5) y con los iconos de las respectivas barras de herramientas (Figura 6). De ellas destacamos unas pocas en cada grupo

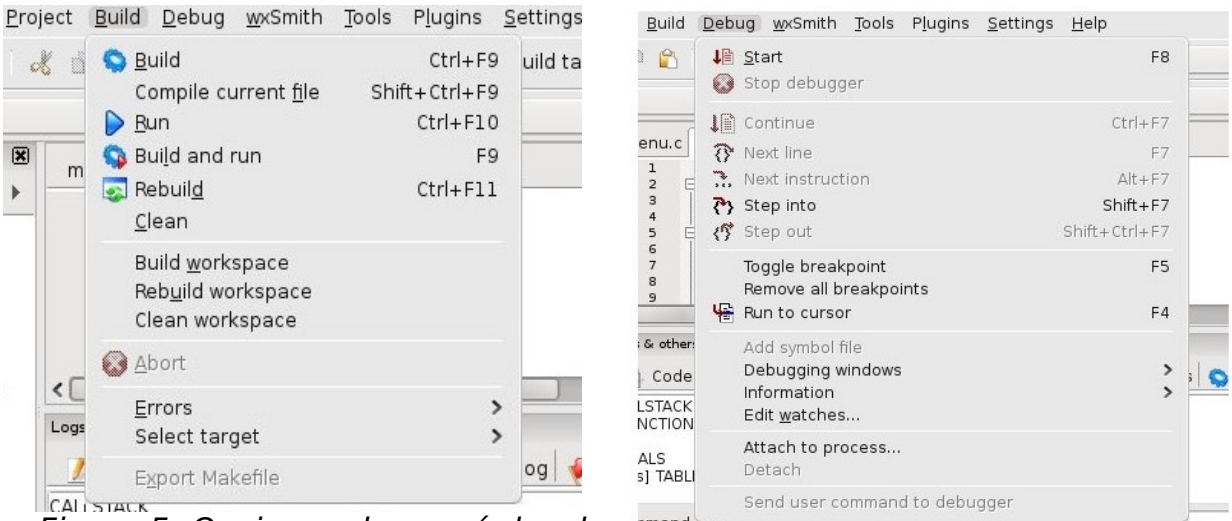

Figura 5: Opciones de menú desplegadas para compilación (izquierda) y depuración (derecha).

Figura 6: Barras de herramientas de compilación (izquierda) y depuración (derecha).

En el menú de compilación:

- **Build**. Es la opción que se seleccionará para volver a crear el nuevo programa ejecutable cada vez que se realice un cambio en alguno de los archivos del proyecto. Recompila sólo los archivos que han sido modificados desde la última construcción del ejecutable.

- **Rebuild**. Recompila el proyecto completo.
- **Select target > Set Active Configuration**. Permite seleccionar la configuración activa: depuración (*Debug*) o distribución (*Release*). Para que pueda depurarse el programa es preciso haber seleccionado la configuración *Debug*.

En el menú de depuración:

- **Start** . Ejecuta el programa en modo de depuración.
- **Continue**. Ejecuta de forma ininterrumpida.
- **Step into.** Ejecuta paso a paso. Si se trata de una llamada a función, el proceso paso a paso continúa dentro de la función llamada.
- **Step out**. Ejecuta el resto del código perteneciente a la función actual, hasta el retorno al punto del programa en que fue llamada.
- **Run to cursor**. Ejecuta hasta la posición del cursor en el archivo actualmente en edición.

5 Conclusión

Se ha abordado en este artículo la base para la creación de proyectos software de pequeña y mediana envergadura caracterizados por utilizar una librería ya existente y desarrollada por otras personas, en nuestro caso OpenCV.

Se ha abordado el trabajo desde la línea de órdenes, así como también utilizando un entorno de desarrollo integrado (IDE) que ofrece una perspectiva más “visual” con Code::Blocks. Ambas opciones se pueden utilizar en cualquier sistema operati GNU/Linux, Mac OS/X y MS/Windows.

En este sentido, proponemos aquí el uso de Code::Blocks como entorno de desarrollo multiplataforma, esto es, nos permitirá cambiar de sistema operativo

vamos a presentar una introducción al desarrollo de aplicaciones en lenguaje C que utilizan funciones documentadas de un API, en concreto OpenCV (véase [1], [4] y [2]) . Aunque es un material de apoyo, está pensando de forma que la realización práctica de lo presentado en este documento conduzca a la obtención de una serie de resultados experimentales: tanto código como ficheros de imágenes de resultados.

6 Bibliografía

[1] “Open Computer Vision, Disponible en <http://sourceforge.net/projects/opencvlibrary/>.

[2] Gary Bradski y Adrian Kaehler, “Learning OpenCV: Computer Vision with the OpenCV”, O'Reilly Press, Octubre 2008.

[3] Code::Blocks, Disponible en http://www.codeblocks.org

[4] “OpenCVWiki” , Disponible en http://opencv.w.illowgarage.com/wiki/Welcome.