Software de obtención y procesamiento de datos a tiempo real con CUDA para sistema de interferometría

Anexo IV: Documentación Técnica de Programación

Proyecto de Fin de Carrera
INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Septiembre de 2012

Autor
Álvaro Sánchez González

Tutor
Guillermo González Talaván

Cotutor
Francisco Valle Brozas

Contenido

Índice de Figuras

1. Introducción

Este anexo contiene toda la información acerca de la implementación de la aplicación.

En un primer apartado se encuentra la información relativa al lenguaje de programación y entorno de desarrollo utilizado.

A continuación se presentan las herramientas y bibliotecas adicionales que se han utilizado para realizar el proyecto. En esta sección se podrán encontrar enlaces a los sitios web de dichas bibliotecas, así como la información de compilación.

El último apartado contiene una breve introducción a Doxygen, herramienta utilizada para generar la documentación del código.

2. Entorno de desarrollo

2.1. Lenguaje de programación

Desde que se determinó el uso de CUDA de NVIDIA como herramienta para acelerar el procesamiento de las imágenes, el leguaje de programación quedó muy restringido, ya que la parte del código que se ejecuta en la GPU, debe estar programada en C.

Esto llevó a que se decidiera programar la aplicación principalmente en C++, con los módulos que se ejecuten en la tarjeta gráfica programados en C.

En todo caso, aunque esta decisión viniera casi obligada por el sistema CUDA, C++ parece el lenguaje más adecuado para programar este tipo de aplicación, debido a su medio-bajo nivel, que permite trabajar con punteros a direcciones de memoria, y controlar toda la ejecución y gestión de memoria byte a byte, pudiendo optimizar el procesado de una forma menos restringida que en otros lenguajes, y con una ejecución definitivamente más rápida que otros lenguajes interpretados.

2.2. Entorno de desarrollo

Figura 1. Logo de Qt

Aunque la aplicación estaba pensada para trabajar en sistemas Windows, era atractiva la idea de usar un entorno de desarrollo de plataforma cruzada para facilitar la potencial portabilidad en un futuro a otras plataformas.

Un kit de desarrollo con estas características muy utilizado es Qt SDK con Qt Creator, entorno multiplataforma que funciona en las tres plataformas más utilizadas: Windows, Linux y Mac OS X.

Sin embargo seguía habiendo una limitación, y es que NVIDIA sólo da soporte para CUDA en Windows para el entorno de desarrollo y el compilador de Microsoft Visual Studio (O Visual C++ Express).

Este problema se salvó, tras varios días de pruebas fallidas, mediante dos medidas:

- Utilizar en Qt el compilador de C++ de Microsoft Visual Studio en vez de MinGW.
- Crear un fichero (cuda.pri) que generara un makefile adecuado para añadir una etapa a la compilación, de forma que los ficheros con extensión ".cu" se compilaran con el compilador especial que NVIDIA proporciona con la SDK de CUDA. La creación de este fichero fue el paso más delicado y costoso en tiempo, ya que su estructura varía de unas versiones a otras de Qt.

Nótese que el uso del compilador de Visual Studio no limita la portabilidad del programa, ya que NVIDIA sí ofrece soporte para MinGW para las otras plataformas.

Por lo que al final se utilizó:

- Entorno de desarrollo Qt Creator, versión 2.4.1.
- Qt SDK, versión 4.8.1, compiladas para Visual Studio 2008.
- Compilador incluido con Microsoft Visual C++ 2008 Express Edition, utilizado a través del IDE Qt Creator.

3. Herramientas y bibliotecas utilizadas

3.1. NVIDIA CUDA®

Esta herramienta permite el procesamiento en paralelo con el multiprocesador de la GPU en vez de con la CPU, lo que supone un enorme incremento de la velocidad a la hora de procesar imágenes.

Figura 2. Logo de NVIDIA CUDA®

Para desarrollar con esta herramienta es necesario:

- Tener una tarjeta gráfica compatible con CUDA (http://developer.nvidia.com/cuda/cuda-gpus), en este caso se ha dispuesto de una GeForce GTX 550 Ti y de una GeForce GT 630M.
- Instalar el kit de desarrollo, cuyas instrucciones de instalación se encuentran en http://developer.nvidia.com/cuda/cuda-downloads. Para el desarrollo del proyecto se usó la versión 4.2 del kit de herramientas.
- Tener actualizado el driver de la tarjeta de forma que el driver sea compatible con CUDA.

Para ejecutar una aplicación con CUDA simplemente es necesario:

- Tener una tarjeta gráfica compatible con CUDA.
- Tener actualizado el driver de la tarjeta de forma que el driver sea compatible con CUDA.

Además el ejecutable debe ir acompañado de los archivos dll: cudart32_42_9.dll y cufft32_42_9.dll, que permiten la detección o no de CUDA.

A efectos de codificación es necesario escribir unas funciones especiales llamadas "kernels" cuyo contenido se ejecuta en la CPU. Estas funciones tienen que aparecer en unos ficheros especiales con extensión ".cu" que deben ser compilados con el compilador proporcionado por la SDK de CUDA.

Los detalles de enlazado con la aplicación se pueden encontrar en el archivo "cuda.pri".

3.2. Biblioteca QwtPlot

Esta biblioteca se utiliza para la representación de gráficos en la aplicación, como son el espectrograma y el gráfico de línea.

Es una biblioteca de código abierto para Qt cuya página de proyecto se encuentra en: http://qwt.sourceforge.net/index.html.

Dado que no distribuyen binarios, es necesario compilarla, en este caso usando el compilador de Microsoft Visual Studio 2008.

La versión de la biblioteca utilizada para el proyecto fue la 6.0.1.

Una vez compilada cuenta tanto con una biblioteca estática como con dos archivos dll (qwt.dll y qwtmath.dll) que deben acompañar al ejecutable.

Se adjunta una versión de la biblioteca 6.0.1 compilada para Visual Studio 2008.

Los detalles de enlazado con la aplicación se pueden encontrar en el archivo "qwt.pri".

3.3. Biblioteca QwtPlot3D

Esta biblioteca se utiliza para la representación de gráficos tridimensionales en la aplicación, como son la superficie 3D y la superficie paramétrica.

Es una biblioteca de código abierto para Qt, basada en OpenGL, encontrándose la página del proyecto en: http://qwtplot3d.sourceforge.net/.

Figura 3. Logo de QwtPlot3D

Dado que tampoco distribuyen binarios, es necesario compilarla, usando también el compilador de Microsoft Visual Studio 2008.

La versión de la biblioteca utilizada para el proyecto fue la 0.2.7.

Una vez compilada cuenta tanto con una biblioteca estática como con un archivo dll (qwtplot3d.dll), además de los relacionados con OpenGL (QtOpenGL4.dll) que deben acompañar al ejecutable.

Se adjunta una versión de la biblioteca 0.2.7 compilada para Visual Studio 2008.

Los detalles de enlazado con la aplicación se pueden encontrar en el archivo "qwt3d.pri".

3.4. API uEye®

Es la API utilizada para la programación de las cámaras de IDS-Imaging (http://www.ids-imaging.de) disponibles en el laboratorio.

Figura 4. Logo de uEye®

Necesita tanto los drivers de la cámara como la API que se pueden descargar de http://www.ueyesetup.com/

La versión de la API y driver utilizados para el proyecto fue la 4.1.

El ejecutable debe ir acompañado del archivo dll "uEye_api.dll".

Los detalles de enlazado con la aplicación se pueden encontrar en el archivo "ueye.pri".

3.5. QuaZIP

Es la biblioteca de código abierto utilizada para la compresión y de-compresión de archivos ZIP (http://quazip.sourceforge.net/).

Está basada en la biblioteca de compresión ZLib por lo que esta biblioteca es necesaria para su compilación.

Dado que tampoco distribuyen binarios, es necesario compilarla usando el compilador de Microsoft Visual Studio 2008.

Una vez compilada cuenta tanto con una biblioteca estática como con un archivo dll (quazip.dll), que debe acompañar al ejecutable.

Se adjunta una versión de la biblioteca 0.4.4 utilizada, compilada para Visual Studio 2008.

Los detalles de enlazado con la aplicación se pueden encontrar en el archivo "quazip.pri".

3.6. MEncoder.exe

Figura 5. Logo de MPlayer

Es una herramienta de MPlayer (http://www.mplayerhq.hu) utilizada para crear los archivos de vídeo en formato AVI con las secuencias de imágenes bajo licencia "GNU General Public License". La versión usada es la 1.1.

Consiste en un archivo ejecutable (mencoder.exe) que acompaña a la aplicación, y que es llamado a través de la clase de Qt: QProcess.

Un ejemplo de comando que hay que ejecutar es:

```
mencoder.exe mf://*.bmp -mf w=640:h=480:fps=24:type=bmp -ovc
lavc lavcopts vcodec=mpeg4:vbitrate=2000 -oac copy -o
salida.avi;
```

Esta aplicación es multiplataforma.

3.7. Converter.exe

Es una herramienta de ImageMagick (http://www.imagemagick.org) utilizada para crear las animaciones en formato GIF a partir de las imágenes de la secuencia bajo licencia "Apache 2.0 license". La versión usada es la 6.7.8-2.

Figura 6. Logo de ImageMagick

Consiste en un archivo ejecutable (converter.exe) junto con un archivo dll (vcomp100.dll) que acompaña a la aplicación, y que es llamado a través de la clase de Qt: QProcess.

Un ejemplo de comando que hay que ejecutar es:

```
convert.exe -delay 100 -loop 0 *.bmp salida.gif
```

Esta aplicación también es multiplataforma.

4. Documentación de clases y funciones construidas

4.1. Introducción

Para la documentación del código, se ha utilizado la herramienta gratuita Doxygen (http://www.stack.nl/~dimitri/doxygen/) que, junto con la aplicación Doxywizard, proporcionada con la herramienta, permite generar en este caso una página web, con toda la documentación de las clases y las funciones construidas.

Figura 7. Logo de Doxygen

Para que esto sea posible, es fundamental añadir los comentarios en el código con el formato indicado para Doxygen:

- Los comentarios deben comenzar por /**. Ejemplo: /** comentario */.
- Se utilizan etiquetas especiales para indicar elementos concretos:
  - @file nombre: Indica el nombre del archivo.
  - @author autor: Indica el autor.
  - @date fecha: Indica la fecha.
  - @brief resumen: Indica un resumen del elemento que se esté documentando.
  - /*! \class clase archivo "archivo": Indica el comienzo de documentación de una clase.
  - @param nombre: Indica la documentación de un parámetro de un método.
  - @return: Indica la documentación del valor de retorno.
  - …

4.2. Documentación de código generada

La documentación del código generada consiste en una página web en la carpeta “AnexoIVApéndiceDocumentaciónTécnica”, cuyo punto de entrada es el archivo “index.html”.

Una captura de pantalla de esta documentación es la siguiente:

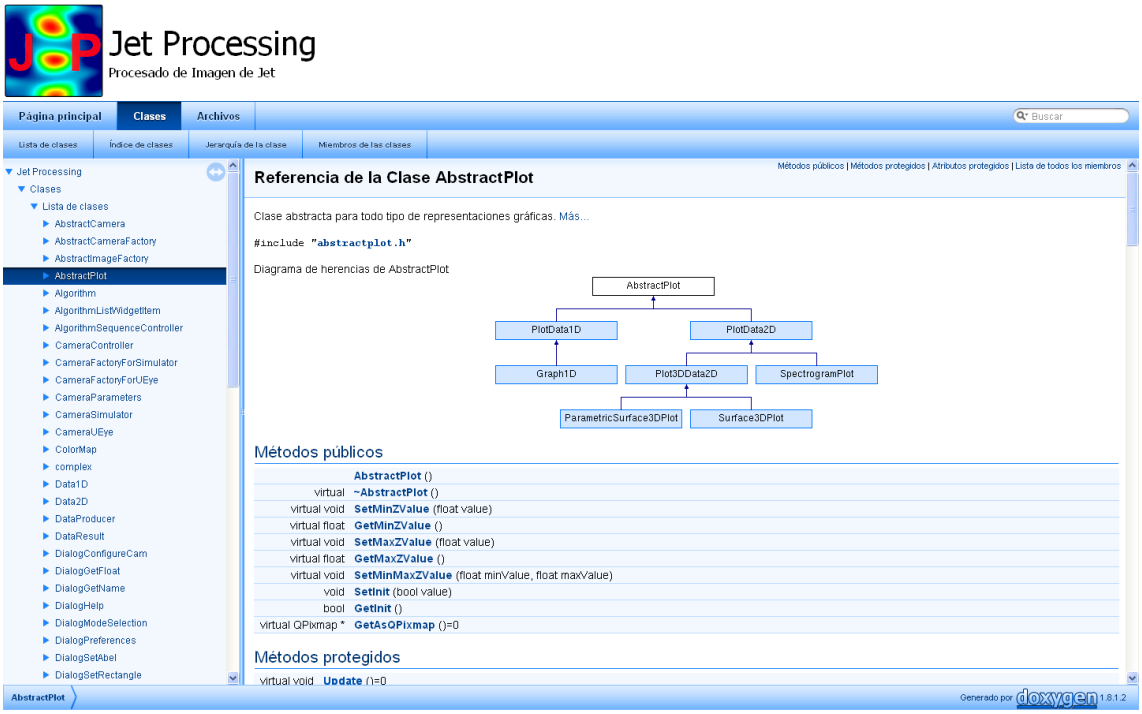

Figura 8. Imagen de la web con la documentación del código

Desde esa web se puede acceder a los comentarios de todas las clases, atributos, métodos, parámetros,… así como buscar en el proyecto, ver diagramas de herencia, o acceder al código donde están implementados los elementos.